

68

MICRO JOURNAL

Australia A \$ 4.75 New Zealand NZ \$ 6.50
 Singapore S \$ 9.45 Hong Kong H \$ 23.50
 Malaysia M \$ 9.45 Sweden 30-SEK

\$2.95_{USA}

OS-9 Atari Amiga Mac S-50

6800 6809 68008 68060 68010 68020 68030

The Magazine for Motorola CPU Devices For Over a Decade!

This Issue:

"C" User Notes p.4

Basically OS-9 p. 14

Logically Speaking p.21

FORTH p. 39

Mac Watch p.43

OS-9 **SE*DOS** Atari Amiga
FLEX Macintosh A User Contributor Journal And Lots More!

VOLUME XI ISSUE I • Devoted to the 68XXX User • January 1989

The Grandfather of "DeskTop Publishing™"

SERVING THE 68XXX USER WORLDWIDE

000422 A/E
 MR. MICKEY FERGUSON
 P.O. BOX 87
 KINGSTON SPRINGS TN 37082

MJ

A6 D14
 A7 D15
 A8 D16
 A9 D17
 A10 D18
 A11 D19
 A12 D20

MC 6809E CPU

40 D16A
 39 D16B
 38 D16C
 37 D16D
 36 D16E
 35 D16F
 34 D16G
 33 D16H
 32 D16I
 31 D16J
 30 D16K
 29 D16L
 28 D16M
 27 D16N
 26 D16O
 25 D16P
 24 D16Q
 23 D16R
 22 D16S
 21 D16T

S

40 D16A
 39 D16B
 38 D16C
 37 D16D
 36 D16E
 35 D16F
 34 D16G
 33 D16H
 32 D16I
 31 D16J
 30 D16K
 29 D16L
 28 D16M
 27 D16N
 26 D16O
 25 D16P
 24 D16Q
 23 D16R
 22 D16S
 21 D16T

MC 68000 CPU

62 D16A
 61 D16B
 60 D16C
 59 D16D
 58 D16E
 57 D16F
 56 D16G
 55 D16H
 54 D16I
 53 D16J
 52 D16K
 51 D16L
 50 D16M
 49 D16N
 48 D16O
 47 D16P
 46 D16Q
 45 D16R
 44 D16S
 43 D16T
 42 D16U
 41 D16V
 40 D16W
 39 D16X
 38 D16Y
 37 D16Z
 36 D16A
 35 D16B
 34 D16C
 33 D16D

YSS D16A
 NMI D16B
 IRQ D16C
 BS D16D
 BA D16E
 VCC D16F
 A0 D16G
 A1 D16H
 A2 D16I
 A3 D16J
 A4 D16K
 A5 D16L
 A6 D16M
 A7 D16N
 A8 D16O
 A9 D16P
 A10 D16Q
 A11 D16R
 A12 D16S

MC 6809 CPU

40 D16A
 39 D16B
 38 D16C
 37 D16D
 36 D16E
 35 D16F
 34 D16G
 33 D16H
 32 D16I
 31 D16J
 30 D16K
 29 D16L
 28 D16M
 27 D16N
 26 D16O
 25 D16P
 24 D16Q
 23 D16R
 22 D16S
 21 D16T



0

WHO DO YOU CALL WHEN YOUR DEBUGGER WON'T DEBUG?



The problem with most real-time operating systems is simple, they're not an integrated solution. You end up dealing with a multitude of suppliers for languages, compilers, debuggers and other important development tools. And when something does go wrong, it can be a frustrating experience trying to straighten out the mess.

Why Not Try the Microware One-Stop Total Solution?

Microware's OS-9 Real-Time Operating System is a total integrated software system, not just a kernel. We offer an extensive set of development tools, languages, I/O and Kernel options. ***And this total integrated solution is entirely designed, built and supported by the same expert Microware team.***

Microware is a registered trademark of Microware Systems Corporation.
OS-9 is a trademark of Microware.
UNIX is a trademark of AT & T.
VAX is a trademark of DEC.

Modularity Lets YOU Choose Just What You Need.

The modular design of OS-9 allows our Operating System to adapt as your requirements change. OS-9 can support a complete spectrum of applications — from embedded ROM-based code in board-level products all the way up to large-scale systems.

Support is Part of the Package.

Microware is proudly setting the industry's standard for customer support. You'll find professional and comprehensive technical documentation and a Customer Hotline staffed by courteous and authoritative software engineers.

So stop messing with simple kernels and independent suppliers. Call Microware today and find out more about the "One-Stop Integrated Solution" with OS-9!

The OS-9 Success Kit

A Total Integrated Solution for Your Next Project

Development Tools:

C Source Level Debugger
Symbolic Debugger
System State Debugger
uMACS Text Editor
Electronic Mail
Communications
Super Shell

Kernel Options:

MMU (Security Protection) Support
Math Coprocessor Support

* Resident or UNIX versions available
** VAX hosted

Languages:

C*
Basic
Pascal
Fortran
Ada**
Assembler*

I/O Options:

SCSI, SASI & SMD Disks
3-, 5-, 8-inch Diskettes
Magnetic Tape
Ethernet - TCP/IP
Arcnet - OS-9/Net

microware® OS-9

Microware Systems Corporation
1900 N.W. 114th Street
Des Moines, Iowa 50322
Phone: 515/224-1929

Western Regional Office
4401 Great America Parkway
Santa Clara, California 95054
Phone: 408/980-0201

Microware Japan Ltd.
41-19 Honcho 4-Chome
Funabashi City
Chiba 273, Japan
Phone: 0474 (22) 1747

Mustang-020 Mustang-08 Benchmarks

IBM AT 7300 Xenix Sys 3
AT&T 7300 UNIX PC 68010
DEC VAX 11/780 UNIX Berkeley 4.2
DEC VAX 11/750
68000 OS-9 68K 8 Mhz
68000 OS-9 68K 10 Mhz
MUSTANG-08 68008 OS-9 68K 10 Mhz
MUSTANG-020 68020 OS-9 68K 16 Mhz
MUSTANG-020 68020 MC68881 UniFLEX 16 Mhz

32 bit Integer	Register Long
9.7	
7.2	4.3
3.6	3.2
5.1	3.2
18.0	9.0
6.5	4.0
9.8	6.3
2.2	6.88
1.8	1.22

Main()

register long i;
for (i=0; i < 999999; ++i);

Estimated MIPS - MUSTANG-020 - 4.5 MIPS,

Burst to 8 - 10 MIPS: Motorola Specs

OS-9

OS-9 Professional Ver	\$850.00
*Includes C Compiler	
Basic OS	300.00
C Compiler	500.00
68000 Disassembler (w/source add: \$100.00)	100.00
Format 77	750.00
Microvare Pascal	500.00
Overpass Pascal	900.00
Syfo-Graph	495.00
Syfo-Spell	195.00
Syfo-Merge	175.00
Syfo-Graph-Spell-Merge	695.00
PAT w/C source	229.00
JUST w/C source	79.95
PAT/JUST Combo	249.30
Sculptor+ (see below)	995.00
COM	125.00

UniFLEX

UniFLEX (68020 ver)	\$450.00
Screen Editor	150.00
Sort-Merge	200.00
BASIC/Pro-Compiler	300.00
C Compiler	350.00
COBOL	750.00
CMODRM w/source	100.00
TMODEM w/source	100.00
X-TALK (see Ad)	99.95
Cross Assembler	50.00
Format 77	450.00
Sculptor+ (see below)	995.00

Standard MUSTANG-020 shipped 12.5 Mhz	
Add for 16.6 Mhz 68020	375.00
Add for 16.6 Mhz 68881	375.00
Add for 20 Mhz 68020/20RAM	750.00

16 Port exp. RS-232	335.00
---------------------	--------

Requires 1 or 2 Adapter Cards below RS232 Adapter	165.00
Each card supports 4 additional ser. ports (total of 16 serial ports supported)	

60 line Parallel I/O card	398.00
Uses 3 68230 Interface/Timer chips, 6 groups of 8 lines each, separate buffer direction control for each group.	

Prototype Board	75.00
are as for both dip and PGA devices & a pre-wired temporary are a up to 512K ORAM.	

SBC AN	475.00
Interface between the system and ARCNET modified token-passing LAN, fiber optics optional - call. LAN software drivers	120.00

Expansion for Motorola I/O Channel Modules	\$195.00
Special for example MUSTANG-020™ system buyers - Sculptor+ \$695.00. SAVE \$300.00	
Software Discounts	

All MUSTANG-020™ system and board buyers are entitled to
discounts on all listed software: 10-70% depending on item. Call or
write for quote. Discounts apply after the sale as well.

Note: Only Professional OS-9 Now Available
(68020 Version) Includes (\$500) C Compiler -
68020 & 68881 Supported - For UPGRADES
Write or Call for Professional OS-9 Upgrade Kit

Mustang Specifications

12.5 Mhz (optional 16.6 Mhz available) MC68020 full 32-bit wide path
32-bit wide data and address buses, non-multiplexed
on chip instruction cache
object code compatible with all 68XXX family processors
enhanced instruction set - math co-processor or interface
68881 math hi-speed floating point co-processor (optional)
direct extension of full 68020 instruction set
full support IEEE P754, draft 10.0
transcendental and other scientific math functions
2 Megabyte of SIP RAM (512 x 32 bit organization)
up to 256K bytes of EPROM (64 x 32 bits)
4 Asynchronous serial I/O ports standard
optional to 20 serial ports
standard RS-232 interface
optional network interface
buffered 8 bit parallel port (1/2 MC68230)
Centronics type pinout
expansion connector for I/O devices
16 bit data path
256 byte address space
2 interrupt inputs
clock and control signals
Motorola I/O Channel Modules
time of day clock/calendar w/battery backup
controller for 2.5 1/4" floppy disk drives
single or double side, single or double density
35 to 80 track selectable (48-96 TPI)
SASI interface
programmable periodic interrupt generator
interrupt rate from micro-seconds to seconds
highly accurate time base (5 PPM)
5 bit sense switch, readable by the CPU
Hardware single-step capability



Don't be misled!
ONLY Data-Comp
delivers the Super
MUSTANG-020

These hi-speed 68020 systems are presently working at NASA, Atomic Energy Commission,
Government Agencies as well as Universities, Business, Labs, and other Critical Applications
Centers, worldwide, where speed, math crunching and multi-user, multi-tasking UNIX C level
V compatibility and low cost is a must.

Only the "PRO" Version
of OS-9 Supported!



This is HEAVY DUTY
Country!

For a limited time we will offer a \$400 trade-in on your
old 68XXX SBC. Must be working properly and
complete with all software, cables and documentation.
Call for more information

Price List:	
Mustang-020 SBC	\$2490.00
Cabinet w/switching PS	\$299.95
5"-80 track floppy DS/DD	\$269.95
Floppy Cable	\$39.95
OS-9 68K Professional Version	\$850.00
C Compiler (\$500 Value)	N/C
Winchester Cable	\$39.95
Winchester Drive 25 Mbyte	\$895.00
Hard Disk Controller	\$395.00
Shipping USA UPS	\$20.00
UniFLEX	Less \$100.00
MC68881 1/2 math processor	Add \$275.00
16.67 Mhz MC68020	\$375.00
16.67 Mhz MC68881	\$375.00
20 Mhz MC68020 Sys	\$750.00
Note all 68881 chips work with 20 Mhz Sys	
Total:	\$5299.80

NEW LOWER PRICES
25 Mbyte HD ~~\$4299.80~~ \$3749.80
85 Mbyte HD ~~\$5748.80~~ \$4548.80

Data-Comp Division



A Decade of Quality Service™
Systems World-Wide

Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, TN 37343

A Member of the CPI Family

68 Micro Journal

10 Years of Dedication to Motorola CPU Users

6800 6809 68000 68010 68020

The Originator of "DeskTop Publishing™"

Publisher
Don Williams Sr.

Executive Editor
Larry Williams

Production Manager
Tom Williams

Office Manager
Joyce Williams

Subscriptions
Cheryl Hodge

Contributing & Associate Editors

Ron Anderson
Ron Voigts
Doug Lurie
Ed Law

Dr. E.M. "Bud" Pass
Art Weller
Dr. Theo Elbert
& Hundreds More of Us

Contents

"C" User Notes	4	Pass
Basically OS-9	14	Voigts
Logically Speaking	21	Jones
FORTH	39	Lurie
Mac-Watch	43	Law
Intelligent Write/Erase MC68HC11		
EEPROM Devices	46	Pinteric
Bit Bucket	52	
Classifieds	56	

68 MICRO JOURNAL

"Contribute Nothing - Expect Nothing" DMW 1986

COMPUTER PUBLISHING, INC.

"Over a Decade of Service"



68 MICRO JOURNAL
Computer Publishing Center
5900 Cassandra Smith Road
PO Box 849
Hixson, TN 37343

Phone (615) 842-4600 Telex 510 600-6630

Copyrighted © 1987 by Computer Publishing, Inc.

68 Micro Journal is the *original* "DeskTop Publishing" product and has continuously published since 1978 using only micro-computers and special "DeskTop" software. Using first a kit built 6800 micro-computer, a modified "ball" typewriter, and "home grown" DeskTop Publishing software. None was commercially available at that time. For over 10 years we have been doing "DeskTop Publishing"! *We originated what has become traditional "DeskTop Publishing"!* Today 68 Micro Journal is acknowledged as the "Grandfather" of "DeskTop Publishing" technology.

68 Micro Journal (ISSN 0194-5025) is published 12 times a year by Computer Publishing Inc. Second Class Postage paid at Hixson, TN, and additional entries. POSTMASTER: send address changes to 68 Micro Journal, POB 849, Hixson, TN 37343.

Subscription Rates

1 Year \$24.50 USA, Canada & Mexico \$34.00 a year.
Others add \$12.00 a year surface, \$48.00 a year Airmail, USA funds. 2 years \$42.50, 3 years \$64.50 plus additional postage for each additional year.

Items or Articles for Publication

Articles submitted for publication must include authors name, address, telephone number, date and a statement that the material is original and the property of the author. Articles submitted should be on diskette, OS-9, SK-DOS, FLEX, Macintosh or MS-DOS. All printed items should be dark type and satisfactory for photo-reproduction. No blue ink! No hand written articles - please! Diagrams o.k.

Please - do not format with spaces any text indents, charts, etc. (source listing o.k.). We will edit in all formatting. Text should fall flush left and use a carriage return only to indicate a paragraph end. Please write for free authors guide.

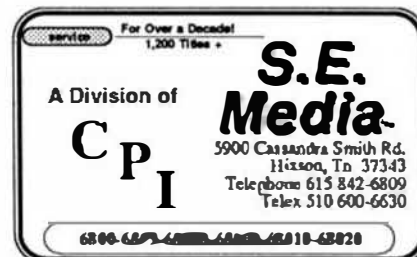
Letters & Advertising Copy

Letters to the Editor should be the original copy, signed! Letters of grip as well as praise are acceptable. *We reserve the right to reject any letter or advertising material, for any reason we deem advisable.* Advertising Rates: Commercial please contact 68 Micro Journal Advertising Department. Classified advertising must be non-commercial. Minimum of \$15.50 for first 15 words. Add \$.60 per word thereafter. No classifieds accepted by telephone.

PAT - JUST

PAT
With 'C' Source
\$229.00

All OS-9
68XXX
Systems



PAT FROM S. E. MEDIA -- A FULL FEATURED SCREEN ORIENTED TEXT EDITOR with all the best of PIE. For those who swore by and loved PIE, this is for YOU! All PIE features & much more! Too many features to list. And if you don't like ours, change or add your own. C source included. Easily configured to your CRT terminal, with special configuration section. No sweat!

68008 - 68000 - 68010 - 68020 OS-9 68K \$229.00

COMBO PAT/JUST

Special \$249.00

JUST

JUST from S. E. MEDIA - - Text formatter written by Ron Anderson; for dot matrix printers, provides many unique features. Output formatted to the display. User configurable for adapting to other printers. Comes set-up for Epson MX80 with Graflex. Up to 10 imbedded printer control commands. Compensates for double width printing. Includes normal line width, page numbering, margin, indent, paragraph, space, vertical skip lines, page length, centering, fill, justification, etc. Use with PAT or any other text editor. The ONLY stand alone text processor for the 68XXX OS-9 68K, that we have seen. And at a very **LOW PRICE!** Order from: S.E. MEDIA - see catalog this issue.

68008 - 68000 - 68010 - 68020
With 'C' source

OS-9 68K
\$79.95

C

*The C Programmers
Reference Source.
Always Right On Target!*

C User Notes

A Tutorial Series

By: Dr. E. M. 'Bud' Pass
1454 Latta Lane N.W.
Conyers, GA 30207
404 483-1717/4570
Computer Systems Consultants

INTRODUCTION

This chapter concludes the presentation of a binary file editor and discusses certain problems with the techniques by which standard C language compilers perform computations

C COMPUTATIONS

As with many other detailed aspects of the C language, the standards leave many specifics of computations to the implementor. It is the programmer's responsibility to be aware of and avoid the common traps.

Since division of integers is always performed in integer mode, the division of a smaller number by a larger number (in absolute value) will always produce zero. Computations must be properly arranged to ensure that incorrect results are not produced because of division truncation.

Thus, the expression $(A * B / C)$ should usually be stated as $((A * B) / C)$, not as $(A * (B / C))$, so that a larger value will be generated by the multiplication operation before the division operation is performed. This will ensure more accurate results.

For example, $((3 * 4) / 5) = 2$, but $(3 * (4 / 5)) = 0$, using integer arithmetic and division truncation, not rounding.

Assuming that the multiplication operation will be performed before the division operation, in the absence of parentheses, will lead to incorrect results on some implementations. Thus, $(3 * 4 / 5)$ may be 0 or 2, depending upon the implementation.

Unfortunately, performing multiplications before division does not necessarily solve the problem. Most C compilers will promote the operations of an expression to the type of the highest subexpression, to maintain the accuracy of the expression.

Thus, if A, B, and C above are 16-bit short integers, the expression will have type short int. Most C compilers will recognize that the product of two 16-bit values requires 32 bits to maintain, and will perform the multiplication and division using 32-bit operations.

However, if the subexpressions are 32-bit long integers, many C compilers will not extend the temporary results to 64 bits, as the hardware for 64-bit computations is usually not available on 32-bit machines and the loss of efficiency required to call subroutines to perform the extended precision arithmetic may be considered excessive.

Most C compilers will not even warn the programmer or user that the computation may be or is actually subject to high-order truncation, leading to potentially disastrous situations in which the numbers being used to test an algorithm, but some combinations of actual inputs cause overflow or division by zero, another common problem.

In the 64-bit case, the programmer must determine that this treatment is always adequate for the specific case, specify double-precision floating-point calculations, or perform the extended precision integer operations assuming 32-bit precision. The latter approach may be easily accomplished for addition and subtraction, but the multiply algorithm is somewhat more complex and the divide algorithm is substantially more complex.

The floating-point approach has the disadvantages of loss of speed and of loss of accuracy, in addition to usually making the object program larger, in the majority of cases, due to the floating-point subroutines. Furthermore, floating-point libraries have historically been prone to subtle types of errors of their own, and this approach should be avoided if not essential to the calculations.

Another point of difference among C compilers relates to sign extension. Most C programmers are aware of the sign extension considerations as they relate to variables of type char, since this varies significantly among compilers.

C programmers are usually less aware of the sign extension rules for such arithmetic and logical operations as right shifts. In fact, K & R specified that sign extensions for right shifts are left to the specific implementation. In a

small sample of several compilers, half extended the sign and half did not. Two of the compilers providing different results produced code for the same hardware, showing that the designers of the compilers had different opinions for sign-extending right shifts.

EXAMPLE C PROGRAM

Following is this month's example C program; it is the second part of zap, as discussed earlier. The remainder is presented in the previous chapter.

```
/* search value in file */
search ()
{
    int bt; /* first byte thereof */
    int chr;
    long first;
    union
    {
        long ll;
        char ss[4];
    } uu;

    if (!gt_val ("Search for ? ", &uu.ll))
        return;
    if (!gt_val ("Start at ? ", &ssstart))
        return;
    if (!gt_val ("Stop at ? ", &ennd))
        return;
    /* temporary using first to hold EOF value */
    first = lseek (fileno(zf), 0L, 2);
    if (!ennd)
    {
        if (f_verbose)
        {
            V_fprintf (stderr, "EOF at ");
            V_fprintf (stderr, deffmt[cur_printmode], first);
            V_fprintf (stderr, "\n");
        }
        ennd = first - cur_type + 1;
    }
    if (ssstart > ennd)
    {
        remark ("start > end", 0L);
        return;
    }
    if (ennd > first)
    {
        if (f_verbose)
            remark ("end > EOF, truncated", 0L);
        ennd = first;
    }
    /* end of using first to hold EOF value */
#ifdef SEARCH_ACTUAL
    if (fseek (zf, ssstart, 0))
    {
        V_fprintf (stderr, "cannot position to ");
        V_fprintf (stderr, deffmt[cur_printmode], ssstart);
        V_fprintf (stderr, "\n");
        return;
    }
#endif
    (void) signal (SIGINT, quit_search);
    /* shift to align */
}
```

```

if (!swab)
{
    if (cur_type == BYTE)
        uu.ss[0] = uu.ss[3];
    else
        if (cur_type == WORD)
        {
            uu.ss[0] = uu.ss[2];
            uu.ss[1] = uu.ss[3];
        }
}
bt = BYTEVAL(uu.ss[0]);
first = sstart;
diddots = interrupted = FALSE;
while (sstart < ennd)
{
    /* print a dot for every 1K processed */
    if (!if_silent && (((first - sstart) & 0x3ff) && sstart > first))
    {
        V_printf (".");
        (void) fflush (stdout);
        diddots = TRUE;
    }
#ifdef SEARCH_ACTUAL
    /* searching the actual values (very slow) */
    if (get_value (sstart) == bt)
    {
        if (cur_type == BYTE)
            foundit (sstart);
        else
            if (get_value (sstart + 1L) == BYTEVAL(uu.ss[1]))
            {
                if (cur_type == WORD)
                    foundit (sstart);
                else
                    if ((get_value (sstart + 2L) == BYTEVAL(uu.ss[2])) &&
                        (get_value (sstart + 3L) == BYTEVAL(uu.ss[3])))
                        foundit (sstart);
            }
    }
    start++;
#else
    /* searching the old contents of the file */
    if (fgetc (zf) == bt)
    {
        if (cur_type == BYTE)          /* looking for byte is easy */
            foundit (sstart);
        else
        {
            chr = fgetc (zf);
            if (chr == BYTEVAL(uu.ss[1]))
            {
                if (cur_type == WORD)
                {
                    foundit (sstart);
                    ungetc (chr, zf);
                }
                else
                {
                    chr = fgetc (zf);
                    if (chr == BYTEVAL(uu.ss[2]))
                    {
                        chr = fgetc (zf);

```

```

        if (chr == BYTEVAL(wu.ss[3]))
            foundit (sstart);
    }
    fseek (zf, sstart + 1L, 0);
}
}
else
    ungetc (chr, zf);
}
}
sstart++;
if (ferror (zf) || feof (zf))
    quit_search ();
#endif
)
/* while (sstart < ennd) */
if (diddots)
    V_printf ("\n");
(void) signal (SIGINT, SIG_DFL);
if (!f_batch && interrupted)
{
    V_printf ("Interrupted at ");
    V_printf (deffmt[cur_printmode], sstart);
    V_printf ("\n");
}
}

/* print verification list */
verify ()
{
    long addr;

    /* display all modifications entered until now. display in portions
     * of cur_printmode. align to lower cur_type boundary */
    for (addr = 0L, tbl_ptr = tbl; tbl_ptr != tbl_free; tbl_ptr++)
        if (tbl_ptr->addr >= addr)
        {
            addr = tbl_ptr->addr & ~(cur_type - 1);
            V_printf ("vfy: ");
            V_printf (deffmt[cur_printmode], addr);
            V_printf ("%c %-7s => ", dp_type[cur_type], pr_val (addr, FALSE));
            V_printf ("%s\n", pr_val (addr, TRUE));
            addr += cur_type;
        }
}

int gt_line (dst, prompt, arg1, arg2, arg3, arg4)
char *dst;
char *prompt;
long arg1;
long arg2;
char arg3;
char *arg4;
{
    if (prompt && !f_silent)
        V_printf (prompt, arg1, arg2, arg3, arg4);
    (void) fflush (stdout);
    if (!gets (dst))
    {
        if (f_batch && !f_silent)
            V_printf ("eof\n");
#ifdef vaxc
        (void) putchar ('\n');
#endif
    }
}

```

```

        return (NULL);
    }
    if (f_batch && !f_silent)
        V_printf ("%s\n", dst);
    if (dst[0] == '^' && dst[1] == '2' && dst[2] == '\0')
        return (FALSE);
    else
        return (TRUE);
}

int gt_val (prompt, l)
char *prompt;
long *l;
{
    *l = 0L;
    while (gt_line (buf, prompt, 0L, 0L, '\0', NULL))
    {
        if (!decod (buf, l))
            return (TRUE);
    }
    return (FALSE);
}

/* display value, using current settings (result is in static area) */
char *pr_val (addr, cur)
long addr;
int cur;          /* 1 = use current, 0 = use previous */
{
    char *cp;
    int i;
    long val;
    static char dst [64];

    last_value = 0;
    if (cur_printmode == ASCII)
    {
        cp = dst;
        for (i = 0; i < cur_type; i++)
        {
            val = getbyte (addr);
            addr++;
            if (val >= ' ' && val < 0177 && val != '\\')
                *cp++ = val;
            else
            {
                *cp++ = '\\';
                switch ((int)BYTEVAL(val))
                {
                    case '\b':
                        *cp++ = 'b';
                        break;
                    case '\n':
                        *cp++ = 'n';
                        break;
                    case '\t':
                        *cp++ = 't';
                        break;
                    case '\f':
                        *cp++ = 'f';
                        break;
                    case '\r':
                        *cp++ = 'r';

```

```

        break;
    case '\\':
        *cp++ = '\\';
        break;
    default:
        V_sprintf (cp, "%o", val);
        while (*cp)
            cp++;
        break;
    }
    *cp++ = ' ';
}
*cp = '\\0';
}
else
{
    val = 0;
    switch (cur_type)
    {
    case BYTE:
        val = getbyte (addr);
        break;
    case WORD:
        if (swab)
        {
            val = getbyte (addr + 1L);
            val = (val << 8) | getbyte (addr);
        }
        else
        {
            val = getbyte (addr);
            val = (val << 8) | getbyte (addr + 1L);
        }
        break;
    case LWORD:
        if (swab)
        {
            val = getbyte (addr + 3L);
            val = (val << 8) | getbyte (addr + 2L);
            val = (val << 8) | getbyte (addr + 1L);
            val = (val << 8) | getbyte (addr);
        }
        else
        {
            val = getbyte (addr);
            val = (val << 8) | getbyte (addr + 1L);
            val = (val << 8) | getbyte (addr + 2L);
            val = (val << 8) | getbyte (addr + 3L);
        }
        break;
    }
    if ((last_value = val) || cur_printmode != OCTAL)
        V_sprintf (dst, deffmt[cur_printmode], val);
    else
        (void) strcpy (dst, "0");
}
return (dst);
}

static char *fmt [] =
{
    "0%05lo 0%05lo%c %-7s ",

```

```

"%6ld %6ld%c %-7s ",
"x%05ld x%05lx%c %-7s ",
"0%05lo 0%05lo%c %-7s "
};

zap (fname)
char *fname;
{
    char chr;                /* scratch */
    int check;               /* checksum value */
    int checkwrite;          /* check for write access */
    int goon;                /* until ^Y is used */
    int i;                   /* scratch */
    int need_head;           /* header toggle */
    long base;               /* base of patching sequence */
    long offset;             /* offset from base */
    long val;                /* holding variable for values */

    checkwrite = TRUE;
    goon = TRUE;
    /* open file */
#ifdef MSDOS
    if (!(zf = fopen (fname, (f_write) ? "rb+" : "rb")))
#else
    if (!(zf = fopen (fname, (f_write) ? "r+" : "r")))
#endif
    {
        cant (fname);
        /* set defaults and allocate table */
        cur_type = BYTE;
        cur_printmode = OCTAL;
        if (!tbl)
        {
            if (!(tbl = (struct ntry *)
                calloc ((unsigned)tbl_max, sizeof (struct ntry))))
                error ("no room for table");
        }
        tbl_cur = tbl_free = tbl;
        prevcnt = 0;          /* reset previous location table */
        /* loop 1 : loop on Base values */
        while (goon && gt_val ("Base ? ", &base))
        {
            /* loop 2 : loop on offset values */
            while (goon && gt_val ("Offset ? ", &offset))
            {
                need_head = TRUE;
                /* loop 3 : loop on patch commands */
                while (goon)
                {
                    if (need_head && !f_silent)
                        V_printf ("Base      Offset  Value  New\n");
                    need_head = FALSE;
                    if (!gt_line (buf, fmt[cur_printmode], base, offset,
                        dp_type[cur_type], pr_val (base + offset, TRUE)))
                        break;
                    switch (buf[0])
                    {
                        {
                            case '\\0':
                                /* close current, advance and open new location */
                                offset += cur_type;
                                break;
                            case '\\':
                                /* re-open current using new type */
                                cur_type = WORD;

```

```

        break;
case '\\':
    /* re-open current using new type */
    cur_type = BYTE;
    break;
case '|':
    /* re-open current using new type */
    cur_type = LWORD;
    break;
case '^':
    if (!buf[1])
    {
        /* close current, back up and open new location */
        offset -= cur_type;
        break;
    }
    if (buf[1] != 'Y' || buf[2])
        break;
    /* FALL THROUGH */
case '\031': /* ^Y */
    goon = FALSE;
    break;
case '>':
    /* goto new location */
    if (!decod (&buf[1], &val))
    {
        push_loc (base + offset);
        offset = buf[1] ? val : last_value;
    }
    break;
case '<':
    /* goto location */
    if (buf[1] == '\0' && prevcnt > 0)
        offset = pop_loc () - base;
    break;
case ';':
    /* change current display mode ... */
    chr = buf[1];
    if (isupper (chr))
        chr = tolower (chr);
    if (chr == 'o')
        cur_printmode = OCTAL;
    else
    if (chr == 'd')
        cur_printmode = DECIMAL;
    else
    if (chr == 'x')
        cur_printmode = HEX;
    else
    if /* ... or store ascii bytes ... */ (chr == 'a')
    {
        cur_printmode = ASCII;
        for (i = 2; chr = buf[i]; i++)
        {
            if (checkwrite && !f_write)
            {
                need_head = TRUE;
                checkwrite = FALSE;
                remark ("no write access", 0L);
            }
            put_byte (base + offset, chr);
            offset++;
        }
    }

```

```

    }
    else
    if /* ... or print modifications ... */ (chr == 'v')
    {
        verify ();
        need_head = TRUE;
    }
    else
    if /* ... or search values */ (chr == 's')
    {
        search ();
        need_head = TRUE;
    }
    break;
default:
    if ((i = decod (buf, &val)) <= 0)
    {
        if (checkwrite && !f_write)
        {
            need_head = TRUE;
            checkwrite = FALSE;
            remark ("no write access", 0L);
        }
        put_value (base + offset, val);
        if (!i)
            offset += cur_type;
        else
            offset -= cur_type;
    }
    }
    /* loop on patch commands */
}
/* loop on offset values */
}
/* loop on base values */
}
/* compute checksum, if requested */
if (f_check || f_sum)
{
    check = 0;
    for (tbl_cur = tbl; tbl_cur != tbl_free; tbl_cur++)
        check ^= (BYTEVAL(tbl_cur->val) |
            (tbl_cur->old << 8) & 0xff00));
    if (f_sum)
    {
        V_printf ("Checksum = ");
        V_printf (deffmt[cur_printmode], check);
        V_printf ("\n");
    }
}
/* apply patches, after checksum verification */
tbl_cur = tbl;
if (f_write)
{
    /* verify checksum */
    if (f_check)
        while (gt_val ("Checksum ? ", &val))
            if (val == check || f_batch)
                break;
    if (!(f_check && val != check))
        for (tbl_cur = tbl; tbl_cur != tbl_free; tbl_cur++)
            ptx_file (tbl_cur->addr, tbl_cur->val);
}

```

```

    if (tbl_cur != tbl_free)
        error ("no modifications made");
    if (!f_silent && f_write && tbl == tbl_free)
        remark ("no modifications requested", 0L);
    /* close file and exit */
    (void) fclose (zf);
}

cant (s)
char *s;
{
    V_fprintf (stderr, "%s: cannot open %s\n", my_name, s);
    exit (1);
}

remark (s, a)
char *s;
long a;
{
    V_fprintf (stderr, "%s: ", my_name);
    V_fprintf (stderr, s, a);
    V_fprintf (stderr, "\n");
}

error (s)
char *s;
{
    V_fprintf (stderr, "%s: %s\n", my_name, s);
    exit (1);
}

swabcheck ()
{
    union
    {
        short a;
        char a[2];
    } u;
    u.s = 0x1357;
#ifdef SWAB
    if SWAB
        if (!(u.a[0] == 0x57 && u.a[1] == 0x13))
            error ("please recompile with \"-DSWAB=0\"");
    #else
        if (!(u.a[0] == 0x13 && u.a[1] == 0x57))
            error ("please recompile with \"-DSWAB=1\"");
    #endif
    #else
        swab = (u.a[0] == 0x57 && u.a[1] == 0x13);
    #endif
}

EOF

```

FOR THOSE WHO NEED TO KNOW

**68 MICRO
JOURNAL™**

Basically OS-9

Dedicated to the serious OS-9 user.
The fastest growing users group world-wide!
6809 - 68020

A Tutorial Series

By: Ron Voigts
2024 Baldwin Court
Glendale Heights, IL 60139

PRACTICE WHAT I PREACH

A few days ago I did the cardinal sin at the keyboard. I put the wrong disk in at the wrong time. I was setting up another disk with some files. The label on the new disk is identical to the one I keep this column on. With nothing but a brief glance I tossed the column's disk into drive /d1 and entered the line:

```
format /d1 "Files Disk" r
```

Well it was all over column was gone! It was a memory. Weeks of work was gone.

The sin was not formatting the wrong disk. I believe it could happen to anyone. Even the best of intentions go bad. The sin was I had no backup disk. And to add to the injury, I had not even made a hardcopy. I know what you are saying, "Ron, what happened to all the preaching about making backups?" Well, all I can say is that I'm guilty. I did not practice what I preached and I paid the price. Now, I'm back to backing up my files.

THE CURRENT STATE OF BASICALLY OS-9

This explanation does not explain why you did not see a column last month. I feel I should explain. To put it bluntly, I have gone to bi-monthly basis. I don't know for how long I will be doing it this way, but for now expect the column every other month.

I should explain some of my reasons for the change. One reason is monthly burn-out. Yes, it is catching up with me. Or maybe it is old age. This column will be 4 years old this coming April and that is a long time. I tip my hat to the old timers writing in the magazine, like Ron Anderson and Dr. E.M. Pass. I certainly we cannot forget Don Williams Sr and his staff. How those guys have done it for 10+ years is beyond me.

Another problem is finding time to get the column out. I really don't where time goes. Einstein said that time is relative. It is like being on a date with a beautiful woman. Time will speed along. The following week while waiting for her to call you, time will creep along. (Female readers please insert "handsome man" for the "beautiful woman".)

I have a few other reasons, but at this time wish not to divulge them. Hopefully as time goes on things will get back to normal and I will once again bring the column back to a monthly basis.

3 YEARS OF PROGRAMS FOR SALE

I have been also at work on a side project for the past few months and now am ready to unveil it. As many of you know this column has been in existence for some 3+ years. It started back in April 1985. This coming April will in fact mark its 4 year anniversary.

There is some history to the column. Originally it was titled Basic OS-9. My task was to write a beginners column on OS-9 helping new comers adjusted and get into the swing of things. I tried to cover as much as I could. I covered programming languages, operating systems, microprocessor theory and stories of my encounters in the world of computer technology. Yes, I also covered OS-9. After some months of writing the column, I received a call from Don Williams. It was a tuesday evening. Don said that he liked the column, saw a strong future for it and thought the name was not adequate. Not only was I covering beginner OS-9, I was coverering everything else. So we changed the column's name to Basically OS-9.

I had a number of philosophies on what I should write about. My main goal was always to enlighten the reader. I thought of it as feeding my readers. In fact one column was even titled "OS-9 Smorgasbord." I wanted to do one thing in every column. I wanted to present a program each time. My belief was that a program was worth far more than all the rambling I could do. I found the computer magazines that only talk and present nothing to do get boring very fast. So almost every column has some piece of related software.

The software covered all the types of languages. I presented programs in Basic09, Pascal, C Language, Assembler, and Kbasic. Some programs were food for thought, while others were stand alone. The programs covered a large variety of things. To name a few: directory alphabitizer, checkbook balancer, a new DATE program, a simultaneous equation solver and new LIST program.

For the past few months, I have been going through my old files finding all the source code from APRIL 1985 through MARCH 1988. I have just finished the task and put them into a 3 Volume set, one for each year of the column. I have also taken care of putting them into a useful form. I have packed, compiled and assembled the code. So, the disks not only contain the source code, they also have the I-Code, P-Code, or 6809 object code. They are ready to run. You will need the Basic09 or Pascal runtime libraries.

Here is a list of what is on the disks.

VOLUME 1 - 1985 TO 1988

1.	APRIL	BSORT	BASIC09
2.	APRIL	SORT	BASIC09
3.	MAY	PAD	BASIC09
4.	JUNE	PDISPLAY	BASIC09
5.	JUNE	FF	ASSEMBLY
6.	JULY	DISKID	BASIC09
7.	AUGUST	DISKLOOK	BASIC09
8.	SEPTEMBER	CRYPT	C
9.	OCTOBER	RABBITS	PASCAL
10.	NOVEMBER	DALPHA	C
11.	DECEMBER	DNC	C
12.	JANUARY	CHECKING	PASCAL
13.	FEBRUARY	DATE	ASSEMBLY
14.	MARCH	FLOOK	BASIC09

VOLUME 2 - 1986 TO 1987

1.	APRIL	DALPHA	C
2.	MAY	MATRIX	C
3.	JUNE	PACK	ASSEMBLY
4.	JUNE	UNPACK	ASSEMBLY
5.	JULY	GETIME	ASSEMBLY
6.	JULY	SORT	C
7.	AUGUST	SYSGEN	BASIC09
8.	SEPTEMBER	ATTACH	ASSEMBLY
9.	OCTOBER	NEWTONS1	KBASIC
10.	OCTOBER	NEWTONS2	KBASIC
11.	NOVEMBER	TREES	PASCAL
12.	JANUARY	DEVICES	KBASIC
13.	FEBRUARY	STRIP	C
14.	MARCH	WRAP	C

VOLUME 3 - 1987 TO 1988

1.	APRIL	HCHECK	BASIC09
2.	MAY	LIST	C
3.	JULY	PCOPY	BASIC09
4.	AUGUST	DOUBLE.V1	C
5.	AUGUST	DOUBLE.V2	C
6.	SEPTEMBER	OPTTEST	C
7.	OCTOBER	DATE	C
8.	NOVEMBER	GETMODDIR.V1	BASIC09
9.	NOVEMBER	GETMODDIR.V2	BASIC09
10.	DECEMBER	EQUATIONS	PASCAL
11.	JANUARY	SYSGO.V1	ASSEMBLY
12.	JANUARY	SYSGO.V2	ASSEMBLY
13.	FEBRUARY	PLIST	C
14.	MARCH	PATH	C

As you can see there is listed over 42 programs. As it turns out this list can be misleading. Some of the Basic09 programs are really a number of procedures inbedded in one file. The program OPTTEST is really a test for a C module which returns the parameters from the input line. Some of them are samples to be used as examples. All the source code is provided on this 3 disk collection. *These can be purchased from South East Media, \$9.95 any single volume or \$24.95 for all 3. All non-USA orders must add an additional \$3.50 for surface mailing or \$6.00 for airmail.*

AND NOW A SELECTION FROM THE MENU

As many of know I am not one for menu driven systems. I prefer to work in an interactive environment. I like the feel of entering some cryptic looking command and seeing some type of result. Perhaps this is what scares of many people. I like the challenge of learning and experimenting.

Unfortunately this scares off many potential computer user. Also businesses cannot really invest the time it takes for a person to learn a system. Therefore, a menu driven system is useful. It is quick and easy to learn. It virtually guides the user through to his final destination.

This month I am presenting a menu system that can be implemented on any OS-9 systems. It may require a few minor adjustments, but nothing major. This is the first version and I can honestly say that it may not be fully debugged. It is a diamond in the rough. It is up to you to polish it.

Listing 1 has the procedure for creating menus. Listing 2 is the actual menu program. Everything is written in Basic09 and should be easy to understand. Let me however explain the theory of how the menus are used.

The directory "/DD/MENU" should be created. If you do not have a /DD, default directory, in your system this can be changed to /D0, /D1, /H0 or whatever. Just remember to change the program accordingly.

The menus are files contained in their directory. Each file consists of a menu title, number of entries and the actual entries. The menu can consist of either commands or lower level menus. Let me explain better by telling the construction of an entry.

The entry consists of 4 parts. First is the category. At this point there are only two types. If a 1 is encountered for category, then the entry is for an OS-9 command or at least something a SHELL can run. If it is 2, then another menu is being called up.

Second is whether a parameter is required. For menus this is not necessary. However, for a command it may be required. Consider the menu line "Directory". It would be executed using the command DIR which can use some type of parameter. Parameter is a boolean which is set either TRUE or FALSE.

Third is the menu line. This is what is displayed. It is what the user sees. It gives a hint of what might happen. It can be like the command shown above. Or it might be another menu like "Word Processing Menu".

Finally is the actual command line. This is what gets processed. It can be command or a menu file found in "/DD/MENU". For the last two examples, they would be "DIR" and "WP_FILE", respectively.

Let us look at an example.

FILE: main TITLE: Main OS-9 Menu ENTRIES: 3 ENTRY 1 CATEGORY: 2 PARAMETER: NO MENU LINE: Word Processing Menu COMMAND: WP_FILE ENTRY 2 CATEGORY: 1 PARAMETER: YES MENU LINE: Directory COMMAND: DIR ENTRY 3 CATEGORY: 1 PARAMETER: NO MENU LINE: Shell COMMAND:

FILE: WP_FILE TITLE: Word Processing Menu ENTRIES: 3 ENTRY 1 CATEGORY: 2 PARAMETER: YES MENU LINE: Standard OS-9 Editor COMMAND: edit ENTRY 2 CATEGORY: 2 PARAMETER: NO MENU LINE: Stylo COMMAND: stylo ENTRY 3 CATEGORY: 2 PARAMETER: NO MENU LINE: DynaStar COMMAND: ds

I think the best way to get to learn how menu works is to type them up, try the example I have given you and see it in action. Please, make changes if you wish. I am considering making a fancier version of this in C or assembly language. Maybe some body out there is willing to give it a try.

That is it for this column. We'll see you next time.

LISTING 1

PROCEDURE make_menu

```

0000      (* *****
001E      (*
0021      (* Name: Make_Menu
0033      (* By: Ron Voigts
0044      (* Date: 3-NOV-88
0055      (*
0058      (* *****
0076      (*
0079      (* Version 1.0          Original
0098      (*
009B      (* *****
00B8      (*
00BB      (*
00BE      (*
00BF      (* Set up complex data type
00DA      TYPE entry_type=category:INTEGER; parameter:BOOLEAN; menu_line:STRING[64]; command:STRING[64]
0101
0102      (* Set up variables
0115      DIM s:STRING[32]
0121      DIM path:BYTE
0128      DIM title:STRING[64]
0134      DIM esize:INTEGER
013B      DIM entry(10):entry_type
0149
014A      (* Get file name
015A      INPUT "Enter file name: ",s
0173      (* Create it in directroy "/dd/menu"
0197      CREATE #path,"/dd/menu"+s:WRITE
01AE      PRINT
01B0
01B1      (* Get menu title
01C2      INPUT "Enter menu title: [64]: ",title
01E2      PUT #path,title
01EC      PRINT
01EE
01EF      (* How many entries?
0203      INPUT "Enter number of entries: [10]: ",esize
022A      PUT #path,esize

```

```

0234
0235      (* Get the the information for each item
025D      FOR i:=1 TO esize
0270          PRINT
0272          PRINT "Entry - ",i
0283          PRINT
0285          INPUT "Enter category: ",entry(i).category
02A5          PRINT
02A7          INPUT "Parameter? (Y/N): ",s
02C2          IF LEFT$(s,1)="Y" OR LEFT$(s,1)="y" THEN
02DD              entry(i).parameter:=TRUE
02EB          ELSE
02EF              entry(i).parameter:=FALSE
02FD          ENDIF
02FF          PRINT
0301          INPUT "Enter menu line: [64]: ",entry(i).menu_line
0328          PRINT
032A          INPUT "Enter command: [32]: ",entry(i).command
034F          PUT #path,entry(i)
035E      NEXT i
0369      CLOSE #path
036F      END
0371

```

LISTING 2

PROCEDURE menu

```

0000      (* *****
001E      (*
0021      (* Name: Menu
002E      (* By: Ron Voigts
003F      (* Date: 3-NOV-88
0050      (*
0053      (* *****
0071      (*
0074      (* Version 1.0          Original
0093      (*
0096      (* *****
00B3      (*
00B6      (* Include following modules:
00D3      (* 1. Get_Menu
00E1      (* 2. Screen
00ED      (* 3. Blanks
00F9      (* 4. Run_Shell
0108      (* 5. Choice
0114
0115      (* Set up complex data type
0130      TYPE entry_type=category:INTEGER; parameter:BOOLEAN; menu_line:STRING[64]; command:STRING[64]
0157
0158      (* Set up variables
016B      DIM entry(10):entry_type
0179      DIM title:STRING[64]
0185      DIM level:INTEGER
018C      DIM file:STRING[64]
0198      DIM esize:INTEGER
019F      DIM i:INTEGER
01A6      DIM menu_stack(8):STRING[64]
01B7
01B8      (* The first file will always be "main"
01DF      file:="main"
01EA
01EB      (* Get the "main" menu

```

```

0201     RUN get_menu(file,title,esize,entry)
021A
021B     (* This is level 1
022D     level:=1
0234
0235     (* Main program loop with exit except CNTRL-C
0262     LOOP
0264
0265         (* Print the screen
0278     RUN screen(title,level,esize,entry)
0291
0292         (* Get user's choice
02A6     RUN get_choice(i)
02B0
02B1     (* If i=0 then we are backing out one level
02DC     IF i=0 THEN
02E8
02E9         (* Can't back up further than level 1
030E     IF level=1 THEN
031A         level:=1
0321     ELSE
0325
0326         (* Else we go back a level and get the menu
0351         level:=level-1
035C         file:=menu_stack(level)
0367     RUN get_menu(file,title,esize,entry)
0380     ENDIF
0382     ELSE
0386
0387         (* Process a direct command
03A2     IF entry(i).category=1 THEN
03B4         RUN run_shell(entry(i))
03C1     ENDIF
03C3
03C4         (* Proces another menu
03DA     IF entry(i).category=2 THEN
03EC         file:=entry(i).command
03FA     RUN get_menu(file,title,esize,entry)
0413         menu_stack(level):=file
041F         level:=level+1
042A     ENDIF
042C
042D         (* Must be a mistake!!
0443     IF entry(i).category<1 OR entry(i).category>2 THEN
0462         PRINT
0464         PRINT "Bad Menu Entry!"
0477         PRINT CHR$(7)
047C     ENDIF
047E     ENDIF
0480     ENDLOOP
0484     END
0486

```

PROCEDURE get_menu

```

0000     (* Read menu file from "/dd/menu"
0022     (* Set up complex data type
003D     TYPE entry_type=category:INTEGER; parameter:BOOLEAN; menu_line
        :STRING[64]; command:STRING[64]
0064
0065     (* Passed arguments
0078     PARAM file:STRING[32]
0084     PARAM title:STRING[64]

```



```

0090      PARAM esize:INTEGER
0097      PARAM entry(10):entry_type
00A5
00A6      (* Set up variables
00B9      DIM i,path:INTEGER
00C4
00C5      (* Get file from "/dd/menu"
00E0      OPEN #path,"/dd/menu/"+file:READ
00F8
00F9      (* Get menu title
010A      GET #path,title
0114
0115      (* Get menu size
0125      GET #path,esize
012F
0130      (* Get menu entries
0143      FOR i:=1 TO esize
0154          GET #path,entry(i)
0162      NEXT i
016D      END
016F

```

PROCEDURE screen

```

0000      (* Print the screen
0013      (* Complex data type
0027      TYPE entry_type=category:INTEGER; parameter:BOOLEAN; menu_line:STRING[64]; command:STRING[64]
004E
004F      (* Passed arguments
0062      PARAM title:STRING[64]
006E      PARAM level:INTEGER
0075      PARAM esize:INTEGER
007C      PARAM entry(10):entry_type
008A
008B      (* Set up the variables
00A2      DIM i:INTEGER
00A9      DIM ssize:INTEGER
00B0
00B1      (* Screen width
00C0      ssize:=64
00C7
00C8      (* Clear screen and home cursor
00E7      PRINT CHR$(12)
00EC      PRINT
00EE      PRINT
00F0
00F1      (* Print first line
0104      RUN blanks((ssize-LEN(title))/2)
0115      PRINT title
011A      PRINT
011C
011D      (* Print second line
0131      PRINT DATE$;
0135      RUN blanks(ssize-20)
0141      PRINT USING "'Level ',I3",level
0155      PRINT
0157      PRINT
0159
015A      (* Print menu choices
016F      FOR i:=1 TO esize
0180          PRINT " ";
0186          PRINT USING "i3"; i;
0192          PRINT ". "; entry(i).menu_line

```

```
01A2     NEXT i
01AD     END
01AF
```

PROCEDURE blanks

```
0000     (* Print specified number of blanks
0023     PARAM i:INTEGER
002A     DIM j:INTEGER
0031     FOR j:=1 TO i
0042         PRINT " ";
0048     NEXT j
0053     END
0055
```

PROCEDURE get_choice

```
0000     (* Gets user's choice returning an integer
002A     PARAM i:INTEGER
0031     PRINT
0033     PRINT
0035     INPUT "Choice: ( 0 to Backup) : ",i
0056     END
0058
```

PROCEDURE run_shell

```
0000     (* Routine to run a command
001B     TYPE entry_type=category:INTEGER; parameter:BOOLEAN; menu_line:STRING[64]; command:STRING[64]
0042     PARAM e:entry_type
004B     DIM s:STRING[64]
0057     PRINT
0059     PRINT "Command: ",e.command
006D     PRINT
006F     IF e.parameter THEN
007B         INPUT "Enter parameter: ",s
0093     ENDIF
0095     SHELL e.command+" "+s
00A5     PRINT
00A7
00A8     (* Built in PAUSE feature
00C1     INPUT "Type <RETURN> to continue!",s
00E3     END
```

+++

FOR THOSE WHO NEED TO KNOW

**68 MICRO
JOURNAL™**

Logically Speaking

Most of you will remember Bob from his series of letters on XBASIC. If you like it or want more, let Bob or us know. We want to give you what you want!

The Mathematical Design of Digital Control Circuits

By: R. Jones
Micronics Research Corp.
33383 Lynn Ave., Abbotsford, B.C.
Canada V2S 1E2
Copyrighted © by R. Jones & CPI

SOLUTIONS TO TEST THIRTEEN

1a.

A	B	C	D	
0	0	0	1	1
0	0	1	0	1
0	1	1	1	3
1	0	0	0	1
1	1	0	1	3
1	1	1	0	3
3	3	3	3	

A	B	C	D	
0	0	0	1	1
0	0	1	0	1
1	0	0	0	1
1	0	1	1	

$$S_2^4 AB'CD$$

(1a)

A	B'	C	D	
0	1	0	1	2
0	1	1	0	2
0	0	1	1	2
1	1	0	0	2
1	0	0	1	2
1	0	1	0	2
3	3	3	3	

FIGURE T13S.1A

1b/**

A	B	C	D	
0	0	0	0	0
0	1	0	1	2
0	1	1	0	2
1	0	0	1	2
1	0	1	0	2
1	1	1	1	4

A	B	C	D	
0	1	0	1	2
0	1	1	0	2
1	0	0	1	2
1	0	1	0	2
2	2	2	2	

A'	B	C	D	
1	0	0	0	1
1	1	0	1	3
1	1	1	0	3
0	0	0	1	1
0	0	1	0	1
0	1	1	1	3
3	3	3	3	

A'	B	C	D	
1	0	0	0	1
0	0	0	1	1
0	0	1	0	1
1	0	1	1	

A'	B'	C	D	
1	1	0	0	2
1	0	0	1	2
1	0	1	0	2
0	1	0	1	2
0	1	1	0	2
0	0	1	1	2
3	3	3	3	

$$S_2^4 A'B'CD$$

(1b)

FIGURE T13S.1B

2.

A	B	C	D	E		A	B	C	D	E		A	B'	C	D	E'	
0	0	0	1	0	1	0	0	0	1	0		0	1	0	1	1	3
0	0	1	0	0	1	0	0	1	0	0		0	1	1	0	1	3
0	0	1	1	1	3	1	0	0	0	0		0	1	1	1	0	3
0	1	1	1	0	3	1	0	1	1	0		0	0	1	1	1	3
1	0	0	0	0	1							1	1	0	0	1	3
1	0	0	1	1	3							1	1	0	1	0	3
1	0	1	0	1	3	S_3^5 ABCDE'						1	1	1	0	0	3
1	1	0	1	0	3	(2)						1	0	0	1	1	3
1	1	1	0	0	3							1	0	1	0	1	3
1	1	1	1	1	5							1	0	1	1	0	3
6	4	6	6	4								6	6	6	6	6	

FIGURE T13S.2

3.

A	B	C	D	E	F		A	B	C	D	E	F		A'	B	C	D	E	F	
0	0	0	1	1	0	2	0	0	0	1	1	0		1	0	0	1	1	0	3
0	0	1	0	0	1	2	0	0	1	0	0	1		1	0	1	0	0	1	3
0	1	0	0	1	0	2	0	1	0	0	1	0		1	1	0	0	1	0	3
0	1	0	1	0	0	2	0	1	0	1	0	0		1	1	0	1	0	0	3
0	1	0	1	1	1	4	1	0	0	0	0	1		1	1	0	1	1	1	5
0	1	1	1	1	0	4	1	0	1	0	0	0		1	1	1	1	1	0	5
1	0	0	0	0	1	2	2	2	2	2	2	2		0	0	0	0	0	1	1
1	0	1	0	0	0	2								0	0	1	0	0	0	1
1	0	1	0	1	1	4								0	0	1	0	1	1	3
1	0	1	1	0	1	4								0	0	1	1	0	1	3
1	1	0	1	1	0	4								0	1	0	1	1	0	3
1	1	1	0	0	1	4								0	1	1	0	0	1	3
6	6	6	6	6	6									6	6	6	6	6	6	

A'	B	C	D	E	F		A'	B	C'	D	E	F'	
0	0	0	0	0	1		1	0	1	1	1	1	5
0	0	1	0	0	0		1	0	0	0	0	0	1
0	0	1	0	0	1		1	1	1	0	1	1	5
							1	1	1	1	0	1	5
							1	1	1	1	1	0	5
							1	1	0	1	1	1	5
							0	0	1	0	0	0	1
							0	0	0	0	0	1	1
							0	0	0	0	1	0	1
							0	1	1	1	1	1	5
							0	1	0	0	0	0	1
							6	6	6	6	6	6	

$S_{1,6}^6$ A'BC'DEF'
(3)

Remember that it's possible you made different choices from me when it came to selecting columns to doubly-complement, but you should at least end up with an equivalent symmetric function.

Mile 17 - heading for Mile 18

Last time we were together I promised that you'd meet the "daddy" of symmetric functions, so here he is, the fellow who's going to enable you to design

ITERATIVE NETWORKS

Symmetric circuits form part of a larger class of circuitry known as "iterative" or "re-iterative" networks, so called because there's a basic pattern to them, which repeats itself. ALL iterative networks (including symmetric) can be designed by the technique to which you'll shortly be introduced, though in the case of the sub-class, symmetric circuits, it's more usual to do so by the methods we've already studied.

Another sub-class of iterative networks is called "positional circuits", and deals with the position, or physical relationship, of the relays in the network, rather than just the number of relays operated. From now on it's to be understood that when we speak of "a circuit of 20 relays ...", or any other number for that matter, they'll be lettered (or numbered) consecutively. A "set" is a group of consecutively lettered (numbered) relays, and it's with sets of different types that we'll be dealing next. I think it goes without saying that there MUST be at least one unoperated relay between one set and another. I mention this only because I've sometimes been asked whether, say, relays 4, 5 and 6 could be regarded as a set, and relays 7, 8 and 9 as another. By definition these constitute ONE set, as they follow consecutively without a break. Got it so far? Good! Now let's look at typical circuit specifications that we may be asked to design.

1. A circuit of 50 relays is to be closed if, and only if, one set of exactly 3 relays is operated, and all other relays are unoperated.

NOTE: It's common practice in switching theory and other related branches of mathematics to use the word "iff" to mean "if, AND ONLY IF ...", so we'll adopt this convention from now on.

2. A circuit of 98 relays is to be closed iff ALL sets contain exactly 3 operated relays, otherwise no output.

3. A circuit of 457 relays is to be closed iff there are exactly 2 sets of relays operated, one set to contain 2 relays and the other 3 relays, but not necessarily in that order.

4. A circuit of 153 relays is to be closed iff there are exactly 3 sets of operated relays, no matter how many relays in each set.

Doesn't example 3 look frightening? 457 relays! Wow!! But there's no need to worry, as I can assure you that the circuit is just as easy to design whether there are 45, 457 or 457983 relays involved. In designing positional iterative networks, we're not concerned with how many relays there are, as our whole attention will be focussed on one "prototype" relay somewhere along the chain. We usually picture our prototype as being somewhere about the middle of the chain, that is if we think of its position at all, though it doesn't have to be so.

By convention we call our prototype relay "X", and instead of thinking about relay contacts, we now think in terms of input-lines bringing information into X about conditions in the lower-numbered relays. We also imagine an equal number of output-lines leaving X, and passing on information to the next relay, Y, about conditions UP TO AND INCLUDING X ITSELF. The output-lines of relay-X are, of course, the input-lines of relay-Y.

ITERATIVE NETWORKS - EXAMPLE 1

As a first step in learning our new design process, let's take spec-1 above and try to design this circuit, beginning with Diagram 88a. On our squared pad, we set up two columns, headed $X=0$ and $X=1$, X being our prototype relay. Immediately to the left of the $X=0$ column we write the code "0", and further out to the left again (under a heading "Relays Operated" we write a second "0". The left-hand 0 is interpreted as "No relays operated so far", and the 0 in the Code column is simply our code for this condition. This is the lowest level, or simplest piece of information possible!

Relays Operated	Code	X		
0	0	0	1	
0	0	0	1	(a)

Relays Operated	Code	X		
0	0	0	1	
0	0	0	1	
1L	1	-	2	(b)

Relays Operated	Code	X		
0	0	0	1	
0	0	0	1	
1L	1	-	2	
2L	2	-	3	(c)
3L	3	4	-	
3NL	4	4	-	

Relays Operated	Code	X		Z
0	0	0	1	
0	0	0	1	
1L	1	-	2	
2L	2	-	3	(d)
3L	3	4	-	1
3NL	4	4	-	1

Relays Operated	Code	X		Z
0	0	0	1	
0	0	0	1	
1L	1	-	2	
2L	2	-	3	(e)
3L, 3NL	3	3	-	1

Diagram 88

Now let's look at the X=0 column of Code-0 in 88a. With information 0 coming in to X, and X itself unoperated, it, too, must obviously pass on the same information to relay-Y (namely, "No relays operated to this point"), which fact we record by entering a 0 in this column. On the other hand, moving to the X=1 column, if relay-X IS operated, it must pass on to relay-Y the information that the relay immediately in front of Y is operated, and that it's the only one so far. Let's give this the code 1, to stand for "one relay operated immediately before you" and insert this code in the X=1 column. Normally we abbreviate this long statement to "1L", for 1-LAST, meaning "one relay operated, and it's the last one in the chain to this point".

Of course, if it's possible for X to pass this info on to Y, it's also possible for similar news to come into X from ITS previous relay. So as we haven't defined what our response will be to Code-1, let's open up a new line, as in 88b, with 1L at the extreme left, and its code "1" in the code column. So now X "knows" that relay-W alone is operated, and if X itself is NOT operated then we obviously have a set of exactly one relay. This is a breach of the specs, so we'll insert a "-" in the X=0 column to indicate a complete blockage of information to relay-Y. This means that power will be cut off at this point!

If X IS operated, however, we now have TWO consecutively operated relays towards the required set of three. At this point in our analysis we have no code for this condition, so let's create code-2 to represent it, and insert 2 in the X=1 column. This signifies that X is passing along to Y the info 2L, meaning "the two last relays in the chain to this point are operated".

And so we proceed to 88c, where I've taken the process to completion, but we'll continue with the discussion, starting at 2L, code-2, and assume we have this info coming in to X, rather than being transmitted to Y. What do we do here? Well, if X=0, a set of two has been created, which again is a breach of specs, so we'll cut off power by inserting a "-" in this column. If X IS operated, we'll have a sequence of 3 relays to this point, so let's code this as 3, to stand for 3L, or "the last 3 relays are operated to this point", and insert 3 in the X=1 column.

This being another new code, we open up line 3L with 3 in the code-column, and analyse the reaction of X if this info were coming in on ITS input-lines. If X is not operated, we've now closed off our set of 3 relays, so let's code this as 4, to stand for 3NL, or "three relays operated, but not the LAST 3", and insert this code 4 in the X=0 column. Alternatively, we could let code 4 stand for 1S, meaning "We've got our set of 3 completed", but for now we'll stick with 3NL. If X IS operated, we've got four relays in our set (a breach of specs), so we'll insert a "-" in the X=1 column to cut off power.

As we have an unanalysed code-4 in our table, we're forced to open up row 3NL, code-4, and ask ourselves what X should do if it's UNoperated. It's obvious that it must pass on this same info to Y, namely, "a complete set of 3 somewhere back down the line", so we'll enter 4 in the X=0 column. And what if X IS operated? Why, it would be starting a second set, and as this is forbidden, we'll enter a "-" in the X=1 column to cut off power again. And here we seem to have ground to a halt, as we didn't create any new codes this time, so the main chore work has been done, and we've got all the specs incorporated into our table 88c.

But not quite all yet, as we still have to code for POWER-OUTPUT! This we do in 88d, by adding a column to the right of our table and heading it "Z", for output. This column has to be completed from the viewpoint of the output device itself - let's say a Light. Now, reading down the "Relays Operated" column, we ask ourselves "Should I switch ON for 'No relays

operated?"" and as the answer is NO, we don't insert anything in the Z-column for this row. Similarly for 1L (meaning "the relay immediately in front of me is operated") and for 2L (meaning "there are 2 relays operated immediately in front of me"). However, the info 3L ("3 relays operated immediately in front of me") or 3NL ("a set of 3 relays operated somewhere down the line") brings the response "YES, I should light up!", so we insert a 1 in column-Z for these two codes. Let's take time out to re-read all that lot before moving on, but, in any case, I'm going to work out some more examples later on, just to be sure you've got it! But first we must see how to go about

ELIMINATING REDUNDANCIES IN ITERATIVE NETWORKS

Just as we've done with other procedures, our last step before drawing the actual circuit is to reduce the number of lines in the table by seeking out, and eliminating, any redundancies which may exist. We'll deal with this subject more fully later on, but for now, to maintain our flow of thought, we'll adopt a very simplified procedure.

First, we can divide our table into two definite sections, one which calls for an output of 0 and one which calls for 1. Redundancies due to equivalences cannot occur between sections, as the outputs also have to be equivalent, just as with sequential flow-tables. So let's start with Group 0, where we see that Line-0 (code 0) has nothing in common with the other two lines of the same group, so we move on to line-1. Be warned that "-" does NOT correspond to a phi, and must NEVER, under any circumstances, be combined with anything but another "-".

Comparing line-1 with line-2, we note that the X=0 column matches. As far as the X=1 column is concerned, these two lines WOULD be equivalent if only 2 and 3 were equivalent as well. Unfortunately, lines 2 and 3 belong to different groups, so it's impossible for them to be equivalent. Therefore, there are no equivalences in Group-0.

Now let's look at Group-1, and right away we see that line-3 is equivalent to line-4 as both X-columns match and also the output. We therefore superimpose lines 3 and 4 (see 88e), calling it line-3, and change all 4s to 3.

DRAWING THE PROTOTYPE FROM THE FINAL TABLE

It only remains now to draw the network of our prototype cell, bearing in mind that there are actually 50 such cells joined together in a chain. The translation of the table into a circuit is an easy task indeed, as all the necessary info is contained in the table, the circuit being drawn in a matter of seconds.

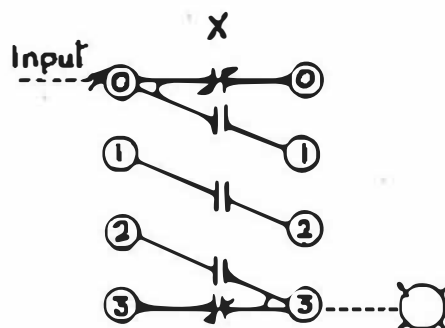


Diagram 89

First, I'll draw for you, in Diagram 89, a single cell corresponding to our prototype. There are two vertical columns of four circles each, labelled 0, 1, 2 and 3, the left-hand column being the input-lines to our prototype, and the right-hand column the output-lines. Power-input occurs at input-line-0 of the first cell of the chain, and as our table shows that power-output occurs only in line-3, the output will therefore be taken off at output-line-3 of the final cell in the chain.

Note the heading "X" above our prototype! Still working from table 88e, we interpret all entries in the X=0 column as being NC-contacts, and those in the X=1 column as NO-contacts. Ignoring the "Relays Operated" column, we read code-0 as saying "Input-line 0 goes to", the "0" in the X=0 column as "output-line-0 via a NC-contact", and the "1" in the X=1 column as "output-line-1 via a NO-contact". The full sentence reads "Input-line-0 goes to output-line-1 via a NC and to output-line-1 via a NO", so this is what we draw at the top part of our prototype.

Code-1 reads as "Input-line-1 goes to nowhere via a NC and to output-line-2 via a NO", so we don't even draw a NC-contact from input-line-1, merely a NO-contact to output-line-2.

Similarly, input-line-2 connects to output-line-3 via a NO-contact only, and finally, input-line-3 connects to output-line-3 via a NC-contact only.

DRAWING THE COMPLETE NETWORK

It's already been mentioned that the complete circuit consists of 50 such cells connected in series, with the output-lines of one cell being the input-lines of the next. Only the first few and the last few cells in the chain will be different. For instance, looking at Diagram 90, we can see that as power comes in on input-line-0 only, input-lines 1, 2 and 3 may be omitted for the first cell. This means that cell-2 will have only input-lines 0 and 1, and so on. Similarly at the output end, output-lines 0, 1 and 2 are unnecessary in the final cell, output-lines 0 and 1 in the last but one, and so on.

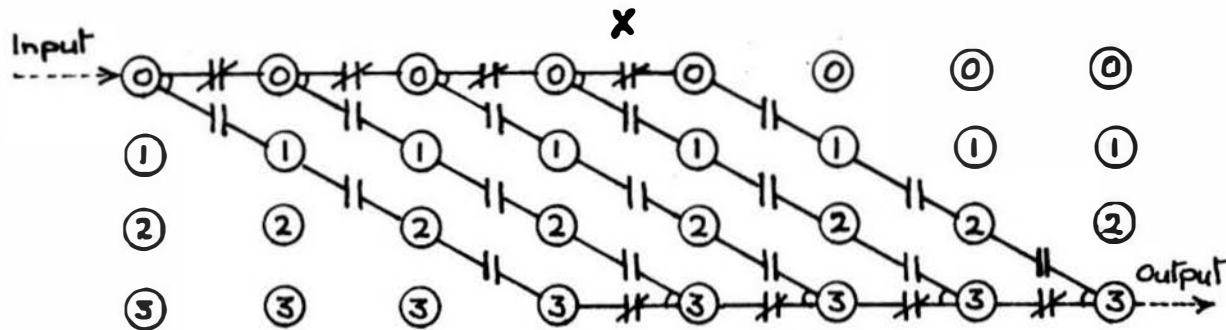


Diagram 90

Diagram 90 is a simplified version of the complete network, showing only one prototype cell X in the centre. Actually, as there are three "odd-ball" cells at either end, there'll be 44 such X-cells in the middle, all exact copies of our full prototype.

A little study of 90 shows that the operation WILL be exactly as per specs! Only if a set of 3 consecutive relays is operated will power be able to flow from the top horizontal line down to the lower, and through to the output via a string of NC-contacts. If any more relays operate, the lower line will become open-circuited and cut off the power, while if less than a set of 3 is operated power will be switched only as far as level-1 or level-2 and be unable to go any further. If no relays are operated at all, power will simply travel along the top rail and terminate in level-0 at the fourth relay from the end.

SOME CHIT-CHAT

With practice, prototype cells can be constructed with very little need for combining lines. For instance, in the example just completed, once we got to constructing the code-3 line of 88c, which signifies that we've JUST completed a set of 3 immediately in front of X, we should realise that from the circuit's viewpoint it matters little whether a fourth relay energises IMMEDIATELY FOLLOWING this set, or whether it's separated by a gap of non-operated relays. In either case, the spec "one set of 3 relays" has been exceeded. Consequently, this particular circuit needn't distinguish between 3L and 3NL, and we could simply have entered "S3" in the "Relays Operated" column (meaning "we have our set of 3"), which is what we ended up with, more or less, in 88e, after having gone through the stage of 88d.

AND SOME "IRICKY" INFO RE ITERATIVE EQUIVALENCES

Before working out some more examples for you, I simply MUST unload this stuff about equivalences! I've already explained the rule that, WITHIN A GROUP, whether Group-0 or Group-1, if two or more lines have EXACTLY the same entries in both X=0 and the X=1 columns, then they're equivalent. The higher-numbered line can thus be eliminated, and all references to this now non-existent line, wherever they occur in the remainder of the table, must be changed to that of the lower-numbered line.

In our worked-out example (see 88d again) we came to the point where we considered combining lines 1 and 2. Our thoughts went something like "1 and 2 WOULD be equivalent if '-' and '-' are (column X=0), which, of course, they are, and also (column X=1) if 2 and 3 are. So let's look at lines 2 and 3! This possibility is ruled out on three counts, (a) '-' and 4 can NEVER be compatible; (b) 3 and '-' can NEVER be compatible; and finally (c) the outputs are incompatible. And so, because we came across an incompatibility in our chain of reasoning, the original thoughts about 1 and 2 are INVALID, and they cannot be equivalent."

But suppose for a moment that line-3 not only had a 0-output, but also had the entries '-' and '1' in the X-columns. You should draw this imaginary chart and follow our new line of thought. Here we go! "The equivalence of 1 and 2 depends on the equivalence of 2 and 3, so let's look at lines 2 and 3. Aha! The two '-'s in column X=0 are equivalent, but column X=1 says that the equivalence of lines 2 and 3 depends on the equivalence of lines 3 and 1, so let's look at lines 1 and 3! But the

equivalence of lines 1 and 3 depends (in column $X=1$) on the equivalence of lines 2 and 3. So here we are, back at our starting-point AND NOWHERE ALONG THE WAY HAVE WE COME ACROSS A NON-EQUIVALENCE. Therefore ALL the pairs we've considered in this chain of reasoning are equivalent, namely 1 and 2, 2 and 3, and 1 and 3." That is to say, all three lines 1, 2 and 3 would be equivalent and could be rolled up into a single line-1.

In sequential flow-tables we regard different lines as being non-equivalent unless we can show them to be equivalent or pseudo-equivalent, but in prototype tables all lines within a group are regarded as being equivalent unless a chain of dependences ends with a non-equivalence somewhere. In other words, in sequential theory we must prove that two or more rows ARE equivalent, otherwise they're not, but in prototype tables we must prove that they're NOT equivalent, otherwise they are.

Some examples of simple equivalences are shown in Diagram 91, so we'll rely on this intuitive approach until we come to a more precise method for dealing with random-input sequential circuits some distance along our journey. Apart from TESTS I doubt that you'll have a REAL NEED to design an iterative network in the meantime.

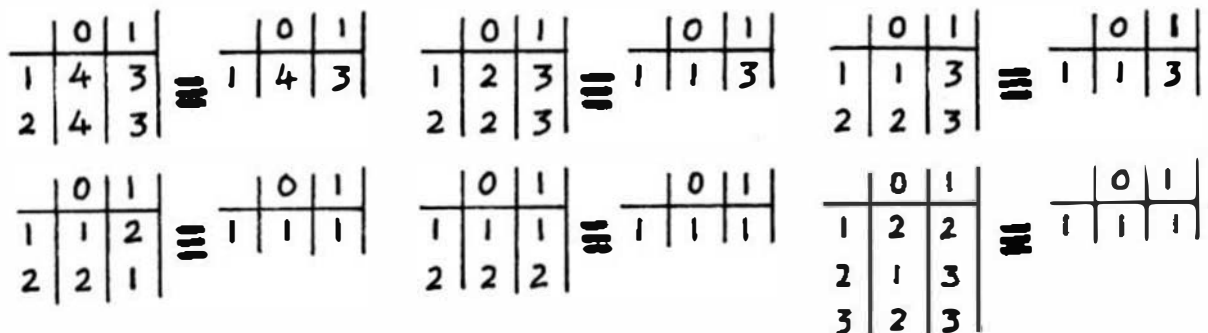


Diagram 91

The triple horizontal bars are read as "is equivalent to". Note in the final example, that although the three rows would be completely incompatible in sequential theory, our thoughts here would be "1 is equivalent to 2 if 2 is equivalent to 1 (column $X=0$), which it IS (as this is the possibility we're considering right now), AND if 2 is equivalent to 3 (column $x=1$), so let's look at lines 2 and 3. 2 and 3 are equivalent, according to column $X=0$, if 1 and 2 are (our original postulation) and if 3 is equivalent to 3 (column $X=1$). So let's look at lines 1 and 2, but, by golly, that's where we started from, and, BECAUSE WE DIDN'T COME ACROSS A NON-EQUIVALENCE ALONG THE WAY, and we've studied lines 1, 2 and 3, then these 3 lines are equivalent. So let's make them all into 1."

TO CONTINUE

And now, to familiarise you with our technique, I'll work out three more examples for you, talking my way through step-by-step. In each case, I'll set out the prototype tables in full, and then justify the entries as I go along, though as an additional exercise you should commence with a blank chart and fill in the entries as we proceed. Off we go with

ITERATIVE NETWORKS - EXAMPLE 2

Design a prototype cell for a circuit of "n" relays which will give an output iff there is exactly one set, consisting of either two or three relays.

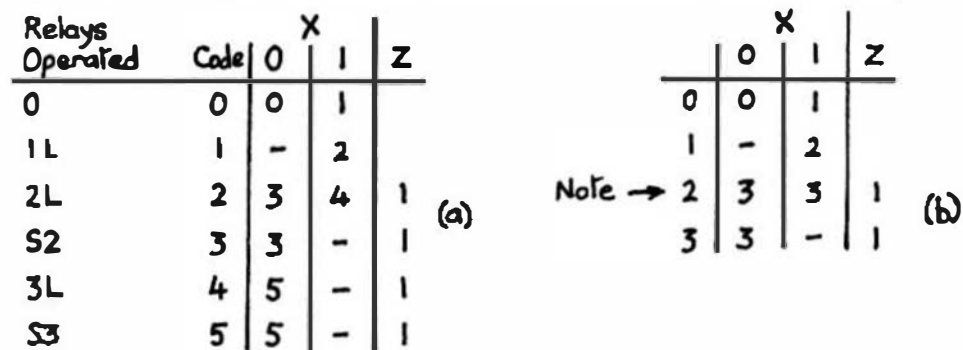
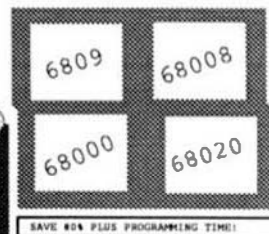


Diagram 92

SCULPTOR

From the world's oldest & largest OS-9 software house!



CUTS PROGRAMMING TIME UP TO 80%
6809/68000-68030 Save 70%

SCULPTOR-a 4GL - Only from S.E. Media at these prices. OS-9 levels one and two (three GIMIX) 6809, all 68XXX OS-9 standard systems. Regular SCULPTOR versions 1.4:6. One of if not the most efficient and easy to develop DBMS type systems running under OS-9! A system of flexible keyed file access that allows extremely fast record and data retrieval, insertion and deletion or other programmed modifications. Access by key or in ascending order, very fast. The system provides automatic menu generation, compilation and report generation. Practically unlimited custom input format and report formatting. A rich set of maintenance and repair utilities. An extremely efficient development environment that cuts most programming approximately 80% in development and debugging! Portable, at source level, to MS-DOS, UNIX and many other languages and systems.

Standard Version: 1.6 6809 - \$1295.00
68000 \$1295.00
68020 \$1990.00

**Due to a "Special One Time" Purchase, We
Are Making This Savings Offer. Quantities
Limited!**

*Once this supply is gone - the price goes
back up!*

System OS-9: 6809/68000-68030

• Regular ~~\$1295.00~~

ONLY

\$295.00

S.E. MEDIA

POB 849

5900 CASSANDRA SMITH ROAD
HIXSON, TN 37343 615 842-4601

+ \$7.50 S&H USA
Overseas - Shipped Air Mail
Collect



SAVE - WHILE SUPPLIES LAST!



Telephone: (615) 842-4600

South East Media

OS-9, UniFLEX, FLEX, SK*DOS SOFTWARE

Telex: 5106006630

!!! Please Specify Your Operating System and Disk Size !!!

SCULPTOR

Full OEM & Dealer Discounts Available!

THE SCULPTOR SYSTEM

Sculptor combines a powerful fourth generation language with an efficient database management system. Programmers currently using traditional languages such as Basic and Cobol will be amazed at what Sculptor does to their productivity. With Sculptor you'll find that what used to take a week can be achieved in just a few hours.

AN ESTABLISHED LEADER

Sculptor was developed by professionals who needed a software development tool with capabilities that were not available in the software market. It was launched in 1981 and since then, with feedback from an ever-increasing customer base, Sculptor has been refined and enhanced to become one of the most adaptable, fast, and above all reliable systems on the market today.

SYSTEM INDEPENDENCE

Sculptor is available on many different machines and for most operating systems, including MS DOS, Unix/Xenix and VMS. The extensive list of supported hardware ranges from small personal computers, through multi-user micros up to large mainframes. Sculptor is constantly being ported to new systems.

APPLICATION PORTABILITY

Mobility of software between different environments is one of Sculptor's major advantages. You can develop applications on a stand-alone PC and -- without any alterations to the programs -- run them on a large multi-user system. For software writers this means that their products can reach a wider marketplace than ever before. It is this system portability, together with high-speed development, that makes Sculptor so appealing to value added resellers, hardware manufacturers and software developers of all kinds.

SPEED AND EFFICIENCY

Sculptor uses a fast and proven indexing technique which provides instant retrieval of data from even the largest of files. Sculptor's fourth generation language is compiled to a compact intermediate code which executes with impressive speed.

INTERNATIONALLY ACCEPTED

By using a simple configuration utility, Sculptor can present information in the language and format that you require. This makes it an ideal product for software development almost anywhere in the world. Australia, the Americas and Europe -- Sculptor is already at work in over 20 countries.

THE PACKAGE

With every development system you receive:

- ☐ A manual that makes sense
- ☐ A periodic newsletter
- ☐ Screen form language
- ☐ Report generator
- ☐ Menu system
- ☐ Query facility
- ☐ Set of utility programs
- ☐ Sample programs

For resale products, the run-time system is available at a nominal cost.

DATA DICTIONARY

Each file may have one or more record types described. Fields may have a name, heading type, size, format and validation list. Field type may be chosen from:

- ☐ alphanumeric
- ☐ integer
- ☐ floating point
- ☐ money
- ☐ date

DATA FILE STRUCTURE

- ☐ Packed, fixed-length records
- ☐ Money stored in lower currency unit
- ☐ Dates stored as integer day numbers

INDEXING TECHNIQUE

Sculptor maintains a B-tree index for each data file. Program logic allows any numbers of alternative indexes to be coded into one other file.

INPUT DATA VALIDATION

Input data may be validated at three levels:

- ☐ automatic by field type
- ☐ validation list in data dictionary
- ☐ program/mvt coded logic

ARITHMETIC OPERATORS

- Unary minus
- * Multiplication
- / Division
- % Remainder
- + Addition
- Subtraction

MAXIMA AND MINIMA

Minimum key length 1 byte
Maximum key length 160 bytes
Minimum record length 3 bytes
Maximum record length 32767 bytes
Maximum fields per record 32767
Maximum records per file 16 million
Maximum files per program 16
Maximum open files
Operating system limit

PROGRAMS

- ☐ Define record layout
- ☐ Create new indexed file
- ☐ Generate standard screen form program
- ☐ Generate standard report program
- ☐ Compile screen form program
- ☐ Compile report program
- ☐ Screen form program editor
- ☐ Report program editor
- ☐ Menu interpreter

RELATIONAL OPERATORS

- = Equal to
- < Less than
- > Greater than
- <= Less than or equal to
- >= Greater than or equal to
- <> Not equal to
- and Logical and
- or Logical or
- ct Contains
- bw Begins with

SPECIAL FEATURES

- ☐ Full date arithmetic
- ☐ Echo suppression for passwords
- ☐ Terminal and printer independence
- ☐ Parameter passing to sub-programs
- ☐ User definable date format

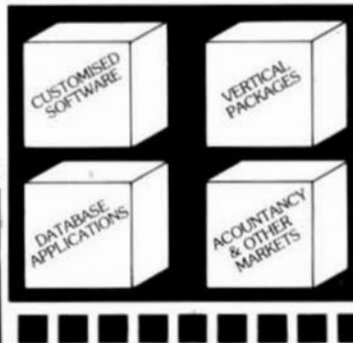
SCREEN FORM LANGUAGE

- ☐ Programmer defined options and logic
- ☐ Multiple files open in one program
- ☐ Default or programmer processing of exception conditions
- ☐ Powerful verbs for input, display and file access
- ☐ Simultaneous display of multiple records
- ☐ Facility to call sub-programs and operating system commands
- ☐ Conditional statements
- ☐ Subroutines
- ☐ Independent of terminal type

Facts
■■■■■

Features
■■■■■■■

**Sculptor for 68020
OS-9 & UniFLEX
\$995**



MUSTANG-020 Users - Ask For Your Special Discount!

MUSTANG-020

***\$1.990 \$398 \$795**

PC/XT/AT/MSDOS

\$695 \$139 \$299

MUSTANG-08

***\$1.295 \$259 \$495**

Call or write for prices on the following systems.

XENIX SYS III & V, MS-NET, UNIX SYS III & V, ATARI OS-9, 68K, UNOS, ULTRIX/VMS (VAX, REGAL), STRIDE, ALTOS, APRICORT, ARETE, ARM, STRONG, BLEASDALE, CHARLES RIVERS, GMX, CONVERG TECH, DEC, CIFER, EQUINOX, GOULD, HP, HONEYWELL, IBM, INTEL, MEGADATA, MOTOROLA, NCR, NIXDORF, N-STAR, OLIVETTI/AT&T, ICL, PERKINS ELMER, PHILLIPS, PIXEL, PLESSEY, PLEXUS, POSITRON, PRIME, SEQUENT, SIEMENS, SWTPC, SYSTIME, TANDY, TORCH, UNISYS, ZYLOG, ETC.

*** For SPECIAL LOW SCULPTOR prices especially for 68010/68XXX OS-9 Systems - See Special Ad this issue. Remember, "When they are gone the price goes back up as above!"**

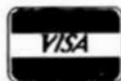
... Sculptor Will Run On Over 100 Other Types of Machines ...

... Call for Pricing ...

!!! Please Specify Your Make of Computer and Operating System !!!

- Full Development Package
- Run Time Only
- C Key File Library

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CD = Color Computer OS-9
CCF = Color Computer FLEX



South East Media
5900 Cassandra Smith Rd. · Hixson, TN. 37343
Telephone: (615) 842-4600 Telex: 5106006630



•• Shipping ••
Add 2% U.S.A. (min. £2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.A.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola. *FLEX and UniFLEX are Trademarks of Technical Systems Consultants. *SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600

South East Media

OS-9, UniFLEX, FLEX, SK-DOS

Telex: 5106006630

ASSEMBLERS

- ASTRUK09** from S.E. Media -- A "Structured Assembler for the 6809" which requires the TSC Macro Assembler.
FLEX, SK-DOS, CCF - \$99.95
- Macro Assembler for TSC - The FLEX, SK-DOS STANDARD Assembler.**
Special -- CCF \$35.00; FLEX, SK-DOS \$50.00
- OSM Extended 6809 Macro Assembler** from Lloyd I/O. -- Provides local labels, Motorola S-records, and Intel Hex records; XREF. Generate OS-9 Memory modules under FLEX, SK-DOS.
FLEX, SK-DOS, CCF, OS-9 \$99.00
- Relocating Assembler/Linking Loader** from TSC. -- Use with many of the C and Pascal Compilers.
FLEX, SK-DOS, CCF \$150.00
- MACE**, by Graham Troit from Windrush Micro Systems -- Co-Resident Editor and Assembler, fast interactive A.L. Programming for small to medium-sized Programs.
FLEX, SK-DOS, CCF - \$75.00
- XMACE** -- MACE w/Cross Assembler for 6800/1/2/3/8
FLEX, SK-DOS, CCF - \$98.00

DISASSEMBLERS

- SUPER SLEUTH** from Computer Systems Consultants Interactive Disassembler; extremely **POWERFUL!** Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly. XREF Generator, Label "Name Changer", and Files of "Standard Label Names" for different Operating Systems.
Color Computer SS-50 Bus (all w/ A.L., Source)
CCD (32K Req'd) Object Only \$49.00
FLEX, SK-DOS \$99.00 - CCF Object Only \$50.00 UniFLEX \$100.00
CCF, with Source \$99.00 OS-9, \$101.00 - CCO, Object Only \$50.00
68010 SUPER SLEUTH - Similar to 8-Bit Version except written in "C".
68010 Disassembler \$100.00 FLEX, UniFLEX, UNIX, XENIX, MS-DOS, SK-DOS, OS-9
OS-9/68K Object Only \$100.00 or with Source \$200.00
- DYNAMITE+** -- Excellent standard "Batch Mode" Disassembler. Includes XREF Generator and "Standard Label" Files. Special OS-9 options with OS-9 Version.
CCF, Object Only \$100.00 - CCO, Object Only \$59.95
FLEX, SK-DOS, Object Only \$100.00 - OS-9, Object Only \$150.00
UniFLEX Object Only \$300.00

CROSS ASSEMBLERS

- CROSS ASSEMBLERS** from Computer System Consultants -- Supports 1802/5, Z-80, 6800/1/2/3/8/11/1/1C11, 6804, 6805/11C05/ 146805, 6809/ 00/01, 6502 family, 8080/5, 8020A/2/35/39/ 40/48/48/49/49/50/ 8748/49, 8031/51/8751, 32000 and 68000/68010 Systems. Assembler and Listing formats same as target CPU's format. Produces machine independent Motorola S-Text. Includes Macro Pre-Processor. Written in "C". 68000 or 6809 *Macintosh, *Atari, FLEX, CCF, UniFLEX, OS-9, XENIX, UNIX, MS-DOS, SK-DOS
any object or source each - \$50.00
any 3 object or source - \$100.00
Set of ALL object \$200.00 - with source \$500.00
- XASM Cross Assemblers** for FLEX, SK-DOS from S.E. MEDIA -- This set of 6800/1/2/3/5/8, 6301, 6502, 8080/5, and Z80 Cross Assemblers uses the familiar TSC Macro Assembler Command Line and Source Code format, Assembler options, etc., in providing code for target CPU's.
Complete set, FLEX, SK-DOS only - \$150.00

CRASMB from LLOYD I/O -- Supports Motorola's, Intel's, Zilog's, and other's CPU syntax for these 8-Bit microprocessors: 6800, 6801, 6303, 6804, 6805, 6809, 6811 (all varieties); 6502, 1802/5, 8048 family, 8051 family, 8080/85, Z8, Z80, and TMS-7000 family. Has MACROS, Local Labels, Label X-REF, Label Length to 30 Chars. Object code formats: Motorola S-Records (text), Intel HEX-Records (text), OS-9 (binary), and FLEX, SK-DOS (binary). Written in Assembler ... e.g. **Very Fast.**

CPU TYPE - Price each:

For:	MOTOROLA	INTEL	OTHER	COMPLETE SET
FLEX9	\$150	\$150	\$150	\$399
SK-DOS	\$150	\$150	\$150	\$399
OS-9/6809	\$150	\$150	\$150	\$399
OS-9/68K	-----	-----	-----	\$432

CRASMB 1632 from LLOYD I/O -- Supports Motorola's 68000, and has same features as the 8 bit version. OS9/68K Object code Format allows this cross assembler to be used in developing your programs for OS-9/68K on your OS-9/6809 computer.

FLEX, SK-DOS, CCF, OS-9/6809 \$249.00

COMMUNICATIONS

- C-MODEM** Telecommunications Program from Computer Systems Consultants, Inc. -- Menu-Driven; supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "Modem7" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".
FLEX, SK-DOS, CCF, OS-9, UniFLEX, UNIX, XENIX, MS-DOS, with Source \$100.00 - without Source \$50.00
- X-TALK** from S.E. Media - X-TALK consists of two disks and a special cable, the hookup enables a 6809 SWTPC computer to dump UniFLEX files directly to the UniFLEX MUSTANG-020. This is the ONLY currently available method to transfer SWTPC 6809 UniFLEX files to a 68000 UniFLEX system. Gimix 6809 users may dump a 6809 UniFLEX file to a 6809 UniFLEX five inch disk and it is readable by the MUSTANG-020. The cable is specially prepared with internal connections to match the non-standard SWTPC SO/9 I/O D625 connectors. A special SWTPC S+ cable set is also available. Users should specify which SWTPC system he/she wishes to communicate with the MUSTANG-020. The X-TALK software is furnished on two disks. One eight inch disk contains S.E. Media modem program C-MODEM (6809) and the other disk is a MUSTANG-020 five inch disk with C-MODEM (68020). Text and binary files may be directly transferred between the two systems. The C-MODEM programs are unaltered and perform as excellent modem programs also. X-TALK can be purchased with or without the special cables, but this special price is available to registered MUSTANG-020 users only.
X-TALK Complete (cable, 2 disks) \$99.95
X-TALK Software (2 disks only) \$69.95
X-TALK with C-MODEM Source \$149.95
- XDATA** from S.E. Media - A COMMUNICATION Package for the UniFLEX Operating System. Use with CP/M, Main Frames, other UniFLEX Systems, etc. Verifies Transmission using checksum or CRC; Re-Transmits bad blocks, etc.
UniFLEX - \$299.99

Availability Legend
O = OS-9, S = SK-DOS
F = FLEX, U = UniFLEX
CCP = Color Computer OS-9
CCF = Color Computer FLEX



South East Media
5900 Cassandra Smith Rd. - Houston, Tx. 77033



** Shipping **
Add 2% U.S.A. (min. \$2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microsoft and Motorola. *FLEX and UniFLEX are Trademarks of Technical Systems Consultants. *SK-DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600

South East Media

Telex: 5106006630

OS-9, UniFLEX, FLEX, SK-DOS

PROGRAMMING LANGUAGES

PL/9 from Windrush Micro Systems -- By Graham Trout. A combination Editor Compiler Debugger. Direct source-to-object compilation delivering fast, compact, re-entrant, ROM-able, PIC. 8 & 16-bit Integers & 6-digit Real numbers for all real-world problems. Direct control over ALL System resources, including interrupts. Comprehensive library support; simple Machine Code interface; step-by-step tracer for instant debugging. 500+ page Manual with tutorial guide.

FLEX, SK-DOS, CCF - \$198.00

PASC from S.E. Media - A FLEX9, SK-DOS Compiler with a definite Pascal "flavor". Anyone with a bit of Pascal experience should be able to begin using PASC to good effect in short order. The PASC package comes complete with three sample programs: ED (a syntax or structure editor), EDITOR (a simple, public domain, screen editor) and CHESS (a simple chess program). The PASC package comes complete with source (written in PASC) and documentation.

FLEX, SK-DOS \$95.00

WHIMSICAL from S.E. MEDIA Now supports Real Numbers. "Structured Programming" WITHOUT losing the Speed and Control of Assembly Language! Single-pass Compiler features unified, user-defined I/O; produces ROMable Code; Procedures and Modules (including pre-compiled Modules); many "Types" up to 32 bit Integers, 6-digit Real Numbers, unlimited sized Arrays (vectors only); Interrupt handling; long Variable Names; Variable Initialization; Include directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. Run-Time subroutines inserted as called during compilation. Normally produces 10% less code than PL/9.

FLEX, SK-DOS and CCF - \$195.00

KANSAS CITY BASIC from S.E. Media - Basic for Color Computer OS-9 with many new commands and sub-functions added. A full implementation of the IF-THEN-ELSE logic is included, allowing nesting to 255 levels. Strings are supported and a subset of the usual string functions such as LEFT\$, RIGHT\$, MID\$, STRING\$, etc. are included. Variables are dynamically allocated. Also included are additional features such as Peek and Poke. A must for any Color Computer user running OS-9.

CoCo OS-9 \$39.95

C Compiler from Windrush Micro Systems by James McCosh. Full C for FLEX, SK-DOS except bit-fields, including an Assembler. Requires the TSC Relocating Assembler if user desires to implement his own Libraries.

FLEX, SK-DOS, CCF - \$295.00

C Compiler from Intrul -- Full C except Doubles and Bit Fields, streamlined for the 6809. Reliable Compiler: FAST, efficient Code. More UNIX Compatible than most.

FLEX, SK-DOS, CCF, OS-9 (Level II ONLY), UniFLEX - \$575.00

PASCAL Compiler from Lucidata -- ISO Based P-Code Compiler.

Designed especially for Microcomputer Systems. Allows linkage to Assembler Code for maximum flexibility.

FLEX, SK-DOS and CCF - \$190.00

OmegaSoft PASCAL from Certified Software -- Extended Pascal for systems and real-time programming.

Native 68000/68020 Compiler, \$575 for base package, options available. For OS-9/68000 and PDOS host system.

6809 Cross Compiler (OS-9/68000 host) \$700 for complete package.

KBASIC from S.E. MEDIA -- A "Native Code" BASIC Compiler which is now Fully TSC XBASIC compatible. The compiler compiles to Assembly Language Source Code. A NEW, streamlined, Assembler is now included allowing the assembly of LARGE Compiled K-BASIC Programs. Conditional assembly reduces Run-time package.

FLEX, SK-DOS, CCF, OS-9 Compiler / Assembler \$99.00

CRUNCH COBOL from S.E. MEDIA -- Supports large subset of ANSI Level 1 COBOL with many of the useful Level 2 features. Full FLEX, SK-DOS File Structures, including Random Files and the ability to process Keyed Files. Segments and link large programs at runtime, or implemented as a set of overlays. The System requires 56K and CAN be run with a single Disk System. A very popular product.

FLEX, SK-DOS, CCF - \$99.95

FORTH from Stearns Electronics -- A CoCo FORTH Programming Language. Tailored to the CoCo! Supplied on Tape, transferable to disk. Written in FAST ML. Many CoCo functions (Graphics, Sound, etc.). Includes an Editor, Trace, etc. Provides CPU Carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. for the "Pro". Excellent "Learning" tool!

Color Computer ONLY - \$58.95

FORTHBUILDER is a stand-alone target compiler (crosscompiler) for producing custom Forth systems and application programs. All of the 83 standard defining words and control structures are recognized by FORTHBUILDER.

FORTHBUILDER is designed to behave as much as possible like a resident Forth interpreter/compiler, so that most of the established techniques for writing Forth code can be used without change.

Like compilers for other languages, FORTHBUILDER can operate in "batch mode".

The compiler recognizes and emulates target names defined by CONSTANT or VARIABLE and is readily extended with "compile-time" definitions to emulate specific target words.

FORTHBUILDER is supplied as an executable command file configured for a specific host system and target processor. Object code produced from the accompanying model source code is royalty-free to licensed users.

FLEX, CCF, SK-DOS - \$99.95

EDITORS & WORD PROCESSING

JUST from S.E. Media -- Text Formatter developed by Ron Anderson; for Dot Matrix Printers, provides many unique features. Output "Formatted" Text to the Display. Use the FPRINT.COM supplied for producing multiple copies of the "Formatted" Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (very useful at other times also, and worth the price of the program by itself). "User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Grafix); up to ten (10) imbedded "Printer Control Commands". Compensates for a "Double Width" printed line. Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc. Use with PAT or any other editor.

* Now supplied as a two disk set:

Disk #1: JUST2.COM object file,

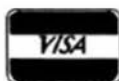
JUST2.TXT PL9 source: FLEX, SK-DOS - CCF

Disk #2: JUSTSC object and source in C:

FLEX, SK-DOS, OS-9, CCF

The JTSC and regular JUST C source are two separate programs. JTSC compiles to a version that expects TSC Word Processor type commands, (.pp .sp .ce etc.) Great for your older text files. The C

Availability Legend
O = OS-9, S = SK-DOS
F = FLEX, U = UniFLEX
CC9 = Color Computer OS-9
CCF = Color Computer FLEX



South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343



** Shipping **
Add 2% U.S.A. (min. \$2.99)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola. *FLEX and UniFLEX are Trademarks of Technical Systems Consultants. *SK-DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600

South East Media

OS-9, UniFLEX, FLEX, SK-DOS

Telex: 5106006630

source compiles to a standard syntax JUST.CMD object file. Using JUST syntax (.p, .u, .y etc.) With all JUST functions plus several additional printer formatting functions. Reference the JUSTSC C source. For those wanting an excellent BUDGET PRICED word processor, with features none of the others have. This is it!

Disk (1) - PL-9 FLEX only- FLEX, SK-DOS & CCF - \$49.95
Disk Set (2) - FLEX, SK-DOS & CCF & OS-9 (C version) - \$69.95
OS-9 68K000 complete with Source - \$79.95

PAT from S.E. Media - A full feature screen oriented TEXT EDITOR with all the best of "PIE"TM. For those who swore by and loved only PIE, this is for you! All PIE features and much more! Too many features to list. And if you don't like these, change or add your own. PL-9 source furnished. "C" source available soon. Easily configured to your CRT, with special config section.

Regular FLEX, SK-DOS \$129.50

* SPECIAL INTRODUCTION OFFER * \$79.95

SPECIAL PAT/JUST COMBO (with source)

FLEX, SK-DOS \$99.95

OS-9 68K Version \$229.00

SPECIAL PAT/JUST COMBO 68K \$249.00

Note: JUST in "C" source available for OS-9

CEDRIC from S.E. Media - A screen oriented TEXT EDITOR with availability of "MENU" aid. Macro definitions, configurable 'permanent' definable MACROS - all standard features and the fastest 'global' functions in the west. A simple, automatic terminal config program makes this a real 'no hassle' product. Only 6K in size, leaving the average system over 165 sectors for text buffer - approx. 14,000 plus of free memory! Extra fine for programming as well as text.

FLEX, SK-DOS \$69.95

BAS-EDIT from S.E. Media - A TSC BASIC or XBASIC screen editor.

Appended to BASIC or XBASIC, BAS-EDIT is transparent to normal BASIC/XBASIC operation. Allows editing while in BASIC/XBASIC. Supports the following functions: OVERLAY, INSERT and DUP LINE. Make editing BASIC/XBASIC programs SIMPLE! A GREAT time and effort saver. Programmers love it! NO more retyping entire lines, etc. Complete with over 25 different CRT terminal configuration overlays.

FLEX, CCF, SK-DOS \$39.95

SCREDITOR III from Windrush Micro Systems -- Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; over 300 pages of Documentation with Tutorial. Features Multi-Column display and editing, "decimal align" columns (AND add them up automatically), multiple keystroke macros, even/odd page headers and footers, imbedded printer control codes, all justifications, "help" support, store common command series on disk, etc. Use supplied "set-up", or remap the keyboard to your needs. Except for proportional printing, this package will DO IT ALL!

6800 or 6809 FLEX, SK-DOS or SSB-DOS, OS-9 - \$175.00

SPELLB "Computer Dictionary" from S.E. Media -- OVER 150,000 words! Look up a word from within your Editor or Word Processor (with the SPH.CMD Utility which operates in the FLEX, SK-DOS UCS). Or check and update the Text after entry; ADD WORDS to the Dictionary, "Flag" questionable words in the Text, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.

FLEX, SK-DOS and CCF - \$129.95

STYLO-GRAPH from Great Plains Computer Co. -- A full-screen oriented WORD PROCESSOR -- (uses the 51 x 24 Display Screens on CoCo FLEX/SK-DOS, or PBJ Wordpak). Full screen display and editing; supports the Daisy Wheel proportional printers.

NEW PRICES 6809 CCF and CCO - \$99.95,

FLEX, SK-DOS or OS-9 - \$179.95, UniFLEX - \$299.95

STYLO-SPELL from Great Plains Computer Co. -- Fast Computer Dictionary. Complements Stylograph.

NEW PRICES 6809 CCF and CCO - \$69.95,

FLEX, SK-DOS or OS-9 - \$99.95, UniFLEX - \$149.95

STYLO-MERGE from Great Plains Computer Co. -- Merge Mailing List to "Form" Letters, Print multiple Files, etc., through Stylo.

NEW PRICES 6809 CCF and CCO - \$59.95,

FLEX, SK-DOS or OS-9 - \$79.95, UniFLEX - \$129.95

STYLO-PAK --- Graph + Spell + Merge Package Deal!!!

FLEX, SK-DOS or OS-9 - \$329.95, UniFLEX - \$549.95

OS-9 68000 \$695.00

DATABASE ACCOUNTING

XDMS from Westchester Applied Business Systems

FOR 6809 FLEX or SK-DOS (\$/8")

Up to 32 groups/fields per record! Up to 12 character file names! Up to 1024 byte records! User defined screen and print control! Process files! Form files! Conditional execution! Process chaining! Upward/Downward file linking! File joining! Random file virtual paging! Built in utilities! Built in text line editor! Fully session oriented! Enhanced format Boldface, Double width, Italics and Underline supported! Written in compact structured assembler! Integrated for FAST execution!

XDMS-IV Data Management System

XDMS-IV is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, but also to process entire files producing customized reports, screen displays and file output. Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and preservation of up to three related files as a "database" on user defined output reports.

POWERFUL COMMANDS!

XDMS-IV combines the functionality of many popular DBMS software systems with a new easy to use command set into a single integrated package. We've included many new features and commands including a set of general file utilities. The processing commands are Input-Process-Output (IPO) which allows almost instant implementation of a process design.

SESSION ORIENTED!

XDMS-IV is session oriented. Enter "XDMS" and you are in instant command of all the features. No more waiting for a command to load in from disk! Many commands are immediate, such as CREATE (file definition), UPDATE (file editor), PURGE and DELETE (utilities). Others are process commands which are used to create a user process which is executed with a RUN command. Either may be entered into a "process" file which is executed by an EXECUTE statement. Processes may execute other processes, or themselves, either conditionally or unconditionally. Menus and screen prompts are easily coded, and entire user applications can be run without ever leaving XDMS-IV

Availability Legend:
O = OS-9, S = SK-DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX



South East Media
5900 Cassandra Smith Rd. - Hixson, TN. 37343



**** Shipping ****
Add 2% (U.S.A. min. \$2.90)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola. *FLEX and UniFLEX are Trademarks of Technical Systems Consultants. *SK-DOS is a Trademark of Star-K Software Systems Corp.

IT'S EASY TO USE!

XDMS-IV keeps data management simple! Rather than design a complex DBMS which hides the true nature of the data, we kept XDMS-IV file oriented. The user view of data relationships is presented in reports and screen output, while the actual data resides in easy to maintain files. This aspect permits customized presentation and reports without complex redefinition of the database files and structure. XDMS-IV may be used for a wide range of applications from simple record management systems (addresses, inventory ...) to integrated database systems (order entry, accounting...)

The possibilities are unlimited...

FOR 6809 FLEX or SK-DOS(5" / 8" Disk) **\$249.95**

UTILITIES

Basic09 XRef from S.E. Media -- This Basic09 Cross Reference Utility is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also includes a Program List Utility which outputs a fast "pretty printed" listing with line numbers. Requires Basic09 or RunB.

OS-9 & CCO object only -- \$39.95; with Source - \$79.95

BTree Routines - Complete set of routines to allow simple implementation of keyed files - for your programs - running under Basic09. A real time saver and should be a part of every serious programmers tool-box.

OS-9 & CCO object only - \$89.95

Lucidata PASCAL UTILITIES (Requires Pascal ver 3)

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

INCLUDE -- Include other Files in a Source Text, including Binary - unlimited nesting.

PROFILER -- provides an Indented, Numbered, "Structogram" of a Pascal Source Text File; view the overall structure of large programs, program integrity, etc. Supplied in Pascal Source Code; requires compilation.

FLEX, SK-DOS, CCF --- EACH 5" - \$40.00, 8" - \$50.00

DUB from S.E. Media -- A UniFLEX BASIC decompiler Re-Create a Source Listing from UniFLEX Compiled basic Programs. Works with ALL Versions of 6809 UniFLEX basic.

UniFLEX - \$219.95

LOW COST PROGRAM KITS from Southeast Media The following kits are available for FLEX, SK-DOS on either 5" or 8" Disk.

- BASIC TOOL-CHEST \$29.95**
BLISTER.CMD: pretty printer
LINXREF.BAS: line cross-referencer
REMPAC.BAS, SPCPAC.BAS, COMPAC.BAS: remove superfluous code
STRIP.BAS: superfluous line-numbers stripper
- FLEX, SK-DOS UTILITIES KIT \$39.99**
CATS. CMD: alphabetically-sorted directory listing
CATD.CMD: date-sorted directory listing
COPYSORT.CMD: file copy, alphabetically
COPYDATE.CMD: file copy, by date-order
FILEDATE.CMD: change file creation date
INFO.CMD (& INFOGMX.CMD): tells disk attributes & contents
RELINK.CMD (& RELINK82): re-orders fragmented free chain
RESQ.CMD: undeletes (recovers) a deleted file
SECTORS.CMD: show sector order in free chain
XL.CMD: super text lister

3. ASSEMBLERS/DISASSEMBLERS UTILITIES \$39.95

LINEFBED.CMD: 'modularise' disassembler output
MATH.CMD: decimal, hex, binary, octal conversions & tables

SKIP.CMD: column stripper

4. WORD - PROCESSOR SUPPORT UTILITIES \$49.95

FULLSTOP.CMD: checks for capitalization

BS1YCIT.BAS (.BAC): Stylo to dot-matrix printer

NECPRI.CMD: Stylo to dot-matrix printer filter code

5. UTILITIES FOR INDEXING \$49.95

MENU.BAS: selects required program from list below

INDEX.BAC: word index

PHRASES.BAC: phrase index

CONTENT.BAC: table of contents

INDXSORT.BAC: fast alphabetic sort routine

FORMATER.BAC: produces a 2-column formatted index

APPEND.BAC: append any number of files

CHAR.BIN: line reader

BASIC09 TOOLS consist of 21 subroutines for Basic09.

6 were written in C Language and the remainder in assembly.

All the routines are compiled down to native machine code which makes them fast and compact.

1. CFILL -- fills a string with characters
2. DPEEK -- Double peek
3. DPOKE -- Double poke
4. FPOS -- Current file position
5. FSIZE -- File size
6. FTRIM -- removes leading spaces from a string
7. GETPR -- returns the current process ID
8. GETOPT -- gets 32 byte option section
9. GETUSR -- gets the user ID
10. GTIME -- gets the time
11. INSERT -- insert a string into another
12. LOWER -- converts a string into lowercase
13. READY -- Checks for available input
14. SETPRIOR -- changes a process priority
15. SETUSR -- changes the user ID
16. SETOPT -- set 32 byte option packet
17. STIME -- sets the time
18. SPACE -- adds spaces to a string
19. SWAP -- swaps any two variables
20. SYSCALL -- system call
21. UPPER -- converts a string to uppercase

For OS-9 - \$44.95 - Includes Source Code

SOFTTOOLS

The following programs are included in object form for immediate application. PL/9 source code available for customization.

READ-ME Complete instructions for initial set-up and operation. Can even be printed out with the included text processor.

CONFIG one time system configuration.

CHANGE changes words, characters, etc. globally to any text type file.

CLEANTXT converts text files to standard FLEX, SK-DOS files.

COMMON compare two text files and reports differences.

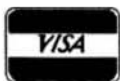
COMPARE another check file that reports mis-matched lines.

CONCAT similar to FLEX, SK-DOS append but can also list files to screen.

DOCUMENT for PL/9 source files. Very useful in examining parameter passing aspects of procedures.

Availability Legends

O = OS-9, S = SK-DOS
F = FLEX, U = UniFLEX
CC9 = Color Computer OS-9
CCF = Color Computer FLEX



South East Media

5900 Cassandra Smith Rd. - Hixson, TN. 37343



** Shipping **

Add 2% U.S.A. (min. \$2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola. *FLEX and UniFLEX are Trademarks of Technical Systems Consultants. *SK-DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600

South East Media

Telex: 5106006630

OS-9, UniFLEX, FLEX, SK-DOS

ECHO echos to either screen or file.

FIND an improved find command with "pattern" matching and wildcards. Very useful.

HEX dumps files in both hex and ASCII.

INCLUDE a file copy program that will accept "includes" of other disk files.

KWIC allows rotating each word, on each line to the beginning. Very useful in a sort program, etc.

LISTDIR a directory listing program. Not super, but better than CAT.

MEMSORT a high-speed text file sorter. Up to 10 fields may be sorted. Very fast. Very useful.

MULTICOL width of page, number of columns may be specified. A MUST!

PAGE similar to LIST but allows for a page header, page width and depth. Adjust for CRT screen or printer as set up by CONFIG. A very smart print driver. Allows printer control commands.

REMOVE a fast file deleter. Careful, no prompts issued. Zap, and its gone!

SCREEN a screen listing utility. Word wraps text to fit screen. Screen depth may be altered at run time.

SORT a super version of MEMSORT. Ascending/descending order, up to 10 keys, case over-ride, sort on nth word and sort on characters if file is small enough, sorts in RAM. If large file, sort is constrained to size of your largest disk capacity.

TPROC a small but nice text formatter. This is a complete formatter and has functions not found in other formatters.

TRANSLIT sorts a file by x keyfields. Checks for duplications. Up to 10 key files may be used.

UNROTATE used with KWIC this program reads an input file and unfolds it a line at a time. If the file has been sorted each word will be presented in sequence.

WC a word count utility. Can count words, characters or lines.

NOTE: this set of utilities consists of 6 5-1/4" disks or 2 8" disks, with source (PL9). 3 5-1/4" disks or 1 8" disk without source.

Complete set SPECIAL INTRO PRICE:

5-1/4" with source FLEX or SK-DOS - \$129.95

without source - \$79.95

8" with source - \$79.95 - without source \$49.95

FULL SCREEN FORMS DISPLAY from Computer Systems Consultants - TSC Extended BASIC program supports any Serial Terminal with Cursor Control or Memory-Mapped Video Displays; substantially extends the capabilities of the Program Designer by providing a table-driven method of describing and using Full Screen Displays.

FLEX, SK-DOS and CCF, UniFLEX - \$25.00, with Source - \$50.00

SOLVE from S.E. Media - OS-9 Levels I and II only. A Symbolic Object/Logic Verification & Examine debugger. Including inline debugging, disassemble and assemble. SOLVE IS THE MOST COMPLETE DEBUGGER we have seen for the 6809 OS-9 series! SOLVE does it all! With a rich selection of monitor, assembler, disassembler, environmental, execution and other miscellaneous commands, SOLVE is the MOST POWERFUL tool-kit item you can own! Yet, SOLVE is simple to use! With complete documentation, a snap! Everyone who has ordered this package has raved! See review - 68 Micro Journal - December 1985. No "blind" debugging here, full screen displays, rich and complete in information presented. Since review in 68 Micro Journal, this is our fastest mover!

Levels I & II only - OS-9 \$69.95

DISK UTILITIES

OS-9 VDisk from S.E. Media -- For Level I only. Use the Extended Memory capability of your SWTPC or Gimix CPU card (or similar format DAT) for FAST Program Compiles, CMD execution, high speed inter-process communications (without pipe buffers), etc. - SAVE that System Memory. Virtual Disk size is variable in 4K increments up to 960K. Some Assembly Required.

Level I OS-9 object \$79.95; with Source \$149.95

O-F from S.E. Media -- Written in BASIC09 (with Source), includes:

REFORMAT, a BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX, SK-DOS Format so it can be used normally by FLEX, SK-DOS; and FLEX, a BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk; user-friendly menu driven. Read the FLEX, SK-DOS Directory, Delete FLEX, SK-DOS Files, Copy both directories, etc. FLEX, SK-DOS users use the special disk just like any other FLEX, SK-DOS disk

OS-9 - 6809/68000 \$79.95

LSORT from S.E. Media - A SORT/MERGE package for OS-9 (Level I & II only). Sorts records with fixed lengths or variable lengths. Allows for either ascending or descending sort. Sorting can be done in either ASCII sequence or alternate collating sequence. Right, left or no justification of data fields available. LSORT includes a full set of comments and error messages.

OS-9 \$85.00

HIER from S.E. Media - HIER is a modern hierarchical storage system for users under FLEX, SK-DOS. It answers the needs of those who have hard disk capabilities on their systems, or many files on one disk - any size. Using HIER a regular (any) FLEX, SK-DOS disk (8" - 5" - hard disk) can have sub-directories. By this method the problems of assigning unique names to files is less burdensome. Different files with the exact same name may be on the same disk, as long as they are in different directories. For the winchester user this becomes a must. Sub-directories are the modern day solution that all current large systems use. Each directory looks to FLEX, SK-DOS like a regular file, except they have the extension '.DIR'. A full set of directory handling programs are included, making the operation of HIER simple and straightforward. A special install package is included to install HIER to your particular version of FLEX, SK-DOS. Some assembly required. Install indicates each byte or reference change needed. Typically - 6 byte changes in source (furnished) and one assembly of HIER is all that is required. No programming required!

FLEX - SK-DOS \$79.95

COPYMULT from S.E. Media -- Copy LARGE Disks to several smaller disks. FLEX, SK-DOS utilities allow the backup of ANY size disk to any SMALLER size diskettes (Hard Disk to Floppies, 8" to 5", etc.) by simply inserting diskettes as requested by COPYMULT. No fooling with directory deletions, etc. COPYMULT.CMD understands normal "copy" syntax and keeps up with files copied by maintaining directories for both host and receiving disk system. Also includes BACKUP.CMD to download any size "random" type file; RESTORE.CMD to restructure copied "random" files for copying, or recopying back to the host system; and FREELINK.CMD as a "bonus" utility that "relinks" the free chain of floppy or hard disk, eliminating fragmentation.

Completely documented Assembly Language Source files included. ALL 4 Programs (FLEX, SK-DOS, 8" or 5") \$99.50

Availability Legend:
O = OS-9, S = SK-DOS
F = FLEX, U = UniFLEX
CC = Color Computer OS-9
CCF = Color Computer FLEX



South East Media
5900 Cassandra Smith Rd. - Hixson, TN. 37343



** Shipping **
Add 2% U.S.A. (min. \$2.90)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola. *FLEX and UniFLEX are Trademarks of Technical Systems Consultants. *SK-DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600

South East Media

OS-9, UniFLEX, FLEX, SK-DOS

Telex: 5106006630

COPYCAT from Lucidata -- *Pascal NOT required.* Allows reading TSC Mini-FLEX, SK-DOS, SSB-DOS68, and Digital Research CP/M Disks while operating under SK-DOS, FLEX1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Also includes some Utilities to help out. Programs supplied in Modular Source Code (Assembly Language) to help solve unusual problems.

FLEX, SK-DOS and CCF 5" - \$50.00 FLEX, SK-DOS 8" - \$65.00

VIRTUAL TERMINAL from S.E. Media - Allows one terminal to do the work of several. The user may start as many as eight tasks on one terminal, under *VIRTUAL TERMINAL* and switch back and forth between tasks at will. No need to exit each one; just jump back and forth. Complete with configuration program. The best way to keep up with those background programs.

6809 OS-9 & CCO - object only - \$49.95

FLEX, SK-DOS DISK UTILITIES from Computer Systems Consultants --

Eight (8) different Assembly Language (with Source Code) FLEX, SK-DOS Utilities for every FLEX, SK-DOS Users Toolbox: Copy a File with CRC Errors; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free-Chain on the Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order). -- PLUS -- Ten X BASIC Programs including: A BASIC Resequencer with EXTRAS over "RENUM" like check for missing label definitions, processes Disk to Disk instead of in Memory, etc. Other programs Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files. A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, X BASIC, and PRECOMPILER BASIC Programs.

ALL Utilities include Source (either BASIC or A.L. Source Code).

FLEX, SK-DOS and CCF - \$50.00

BASIC Utilities ONLY for UniFLEX -- \$30.00

MS-DOS to FLEX Transfer Utilities to OS-9 For 68XXX and CCO-9 Systems Now READ - WRITE - DIR - DUMP - EXPLORE FLEX & MS-DOS Disk. These Utilities come with a rich set of options allowing the transfer of text type files from/to FLEX & MS-DOS disks. *CoCo systems require the D.P. Johnson SDISK utilities and OS-9 and two drives of which one must be a "host" floppy.

**CoCo Version: \$69.95*

68XXX Version \$99.95

MISCELLANEOUS

TABULA RASA SPREADSHEET from Computer Systems Consultants --

TABULA RASA is similar to DESKTOP/PLAN; provides use of tabular computation schemes used for analysis of business, sales, and economic conditions. Menu-driven; extensive report-generation capabilities. Requires TSC's Extended BASIC.

FLEX, SK-DOS and CCF, UniFLEX - \$50.00, with Source - \$100.00

DYNACALC -- Electronic Spread Sheet for the 6809 and 68000.

*UniFLEX - \$395.00, FLEX, SK-DOS, OS-9 and SPECIAL CCF - \$250.00
OS-9 68K - \$299.00*

FULL SCREEN INVENTORY/MRP from Computer Systems Consultants Use the Full Screen Inventory System/Materials Requirement Planning for maintaining inventories. Keeps item field file in alphabetical order for easier inquiry. Locate and/or print records matching partial or complete item, description, vendor, or attributes; find backorder or below stock levels. Print-outs in item or vendor order. MRP capability for the maintenance and analysis of Hierarchical assemblies of items in the inventory file. Requires TSC's Extended BASIC.

FLEX, SK-DOS and CCF, UniFLEX - \$50.00, with Source - \$100.00

FULL SCREEN MAILING LIST from Computer Systems Consultants --

The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Locate all records matching on partial or complete name, city, state, zip, or attributes for Listings or Labels, etc. Requires TSC's Extended BASIC.

FLEX, SK-DOS and CCF, UniFLEX - \$50.00, with Source - \$100.00

DIET-TRAC Forecaster from S.E. Media -- An X BASIC program that plans

a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate. Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual. Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. Provides number of days and daily calendar after weight goal and caloric plan is determined.

FLEX, SK-DOS - \$59.95, UniFLEX - \$89.95

GAMES

RAPIER - 6809 Chess Program from S.E. Media -- Requires FLEX, SK-DOS and Displays on Any Type Terminal. Features: Four levels of play. Swap side. Point scoring system. Two display boards. Change skill level. Solve Checkmate problems in 1-2-3-4 moves. Make move and swap sides. Play white or black. This is one of the strongest CHES programs running on any microcomputer, estimated USCF Rating 1600+ (better than most 'club' players at higher levels)

FLEX, SK-DOS and CCF - \$79.95

NEW

MS-DOS/FLEX Transfer Utilities For 68XXX and CoCo* OS-9 Systems.

Now Read, Write, DIR, Dump and Explore FLEX & MS-DOS Disks. Supplied with a rich set of options to explore and transfer text type files from/to FLEX and MS-DOS disks. *CoCo OS-9 requires SDISK utilities & two floppy drives.

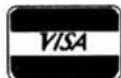
CCO \$69.95 68XXX OS-9 \$99.95

MS-DOS and Macintosh Software at Discounted Prices

"Call for prices, it'll be worth the savings."

(615) 842-4600

Availability Legends
O = OS-9, S = SK-DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX



South East Media
5900 Cassandra Smith Rd. - Hixson, TN. 37343



** Shipping **
Add 7% U.S.A. (min. \$2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola. *FLEX and UniFLEX are Trademarks of Technical Systems Consultants. *SK-DOS is a Trademark of Star-K Software Systems Corp.

First, we draw up a blank table, with all headers, including Z, and start off with 0 in both the "Relays Operated" and "Code" columns. We'll study X's reaction to the info "No relays operated so far". If X=0 there are still no relays operated, so we'll pass on Code-0 to Y, but if X=1 we record 1 to signify 1L ("One relay operated so far").

The creation of code-1 forces us to open up line-1, where, if X=0, a set of 1 is formed, and as this is a breach of specs we'll cut off power with a "-". But if X=1, we have a sequence of 2 so far, so we'll record 2 to signify 2L.

Now we have a code-2 to analyse! If X=0 a complete set of 2 has been formed, so we'll enter a 3 to signify S2 (a set of 2), but if X=1 we now have 3 in sequence, so we'll enter 4 to signify 3L. Note that we now have TWO undefined codes, namely 3 and 4, so we open up lines 3 and 4! Let's do code-3 first.

If X=0, the code S2 still applies and we enter a 3 in this column, but if X=1 we'll be starting a second set (forbidden), so we cut off power. Then to line-4.

In this line, if X=0 we've completed a set of three, which we'll indicate with code-5 to stand for S3 (a set of 3), as this is a new situation, but if X=1 we've "gone over the top" with 4 relays, so we cut off power with a "-". This line forces us to open up line-5, where, if X=0, the situation is still S3, so we enter a 5 in this column, but if X=1 then, obviously, a second set is under way, so once more we'll cut off power.

No new code-numbers means we've come to the end of our design, so let's switch our viewpoint to that of the output, by checking Z against the "Relays Operated" column. Zero relays or one relay immediately before Z certainly don't call for an output, but either 2 relays immediately in front or a set of 2 "down the line somewhere" indicate a need for a "1" in the Z-column, as does 3 relays immediately in front, or a set of 3 down the line.

Equivalences, here we come!! There are only two lines in Group-0, but the "-" in line-1 right away makes them incompatible. So let's study Group-1. Line-2 is incompatible because a 4 CANNOT be combined with a "-". Lines 4 and 5 are certainly compatible, but how about line-3? Let's compare it with line-4, and say "Lines 3 and 4 would match if only 3 and 5 were compatible. Now let's compare 3 and 5! Lines 3 and 5 would match if only 3 and 5 were compatible. (see column X=0 in all cases). We can't go any further, and as we haven't come across an INCOMPATIBILITY, then all three are equivalent, so we'll roll up 3, 4 and 5 into a new line-3, and make all 4s and 5s into 3. We could alternatively have gone the route of rolling-up line-5 into Line-4, as they're definitely equivalent, and making all 5s into 4, then checking lines 3 and 4 by saying "Lines 3 and 4 would match if only 3 and 4 were equivalent. End of run, no incompatibilities, therefore they are so!"

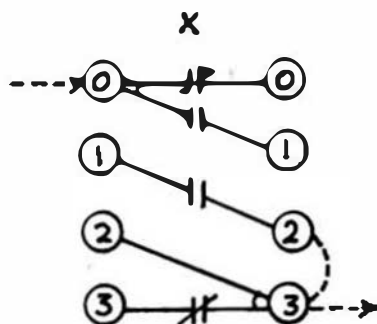


Diagram 93

Either way, we end up with Diagram 92b, where you'll notice I've drawn your attention to Line-2. This is a special case, because both columns of X contain the same 3-entry, and in terms of relay contacts means that input-line-2 is connected to output-line-3 through both a NC and a NO-contact. In other words, whether X is operated or not, input-line-2 will be connected to output-line-3, and we can save ourselves these two contacts by making a direct connection instead, as I've shown in the prototype of Diagram 93.

NEW PROTOTYPING RULE

And so we have a new rule for prototyping

If the same code occurs in both columns X=0 and X=1 in any one line, a direct connection will be made between the input and output lines concerned.

Note that power is taken off at both output lines 2 AND 3 as column-Z has a 1-entry in BOTH these lines. Are you beginning to get the hang of it now? Let's do another example to give you a better picture.

ITERATIVE NETWORKS - EXAMPLE 3

Design a prototype cell for a circuit of "n" relays which will give an output iff there is exactly one set, which **MUST** contain an odd number of relays.

That's certainly different from what we've encountered so far, but as we thrive on variety let's begin with all column headings marked out, and the codings for line-0 inserted, as shown in Diagram 94a. So ... if code-0 comes into X and X=0 the same code goes out, but if X=1 we'll send out code-1 to indicate "Odd/L" (meaning "an odd number immediately in front of you").

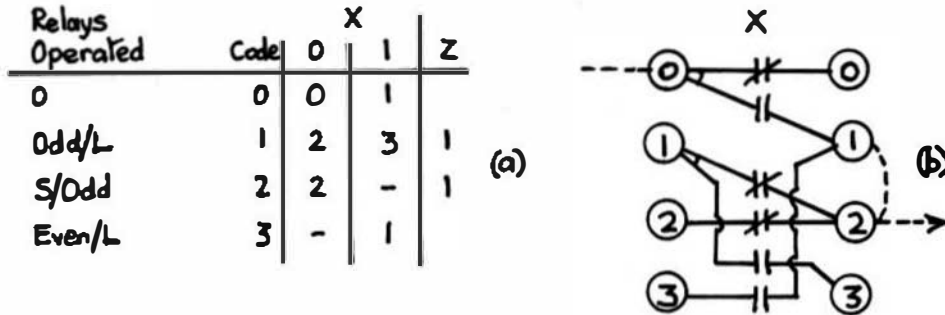


Diagram 94

In line-1, if X=0, we have our odd set, which we'll identify with code-2 and S/odd. If X=1 we have an even set SO FAR, which we'll code as 3 and Even/L. This, of course, means opening up two new lines, 2 and 3. Considering line-2, if X=0 the state of affairs remains unchanged at S/odd, but if X=1 we're starting a second set, so let's cut off the power. And in line-3, with an even-numbered string of relays so far, if X=0 we've formed an even SET, so we'll cut off power here too. But, if X=1, we're back to an odd-numbered string so far, so we'll go back to line-1! Agreed? Finally, looking at things from Z's viewpoint, we'll insert 1s in both lines 1 and 2. There are no equivalences (we must be getting better at this sort of thing!), so we draw the prototype cell of Diagram 94b directly from 94a. The input power comes in at Line-0, and is taken off at output lines 1 and 2 of the final cell, just as it came off at lines 2 and 3 in Example-2. One more example to wrap this whole thing up.

ITERATIVE NETWORKS - EXAMPLE 4

Design a prototype cell for a network of "n" relays which will give an output if there are **EXACTLY** two sets of any size whatsoever.

This looks interesting, so let's go for it! But first note that we're concerned with the number of SETS this time, not the number of RELAYS, so we head our info-column in Diagram 95a with "Sets Operated". Commencing on Line-0, if X=0 the same code goes out, and if X=1 we have the start of Set-1, which we'll call code-1. In line-1, if X=0 we've completed this set, so we'll use code 2 for 1S ("one set completed"), but if X=1 we're still working on our first set, so we insert code-1 here. Now for line-2! If X=0 we still have one set down the line (code-2), but if X=1 we're now starting on our second set. Let's code this as 3 and "1S + 1L", meaning, of course, "one set complete, plus a second starting immediately in front of you". Moving to Line-3, if X=0 we've formed our two sets, which we code as 4, and 2S, whose meaning is quite clear. However, if X=1 we're merely enlarging the size of this second set so we'll stick with code-3. We still have to analyse line-4, where, if X=0, the situation is unchanged at code-4, but if X=1 we're beginning work on a third set (forbidden), so we cut off power. And here the table naturally grinds to a halt.

The output-Z is only interested in two complete sets, so we'll insert 1s in lines 3 and 4. Again there are no equivalences, so the prototype cell shown in Diagram 95b can be drawn directly from 95a.

This concludes the **NORMAL** technique for constructing iterative networks, though next time we'll take a quick look at a more advanced method (not too often used) which **MAY** reduce the number of contacts in a prototype cell, but certainly shouldn't increase them. So now you can all try your hand at **TEST FOURTEEN-A** coming up! **FOURTEEN-B** will unfold next time.

TEST FOURTEEN-A

Design the prototype cell for a circuit of "n" relays which will give an output

- (a) iff there is only ONE set, consisting of either ONE or THREE relays.
- (b) iff ALL sets consist of ONE or THREE relays, and there's at least one set.
- (c) iff ALL sets consist of ONE or THREE relays, OR if no relays are operated.
- (d) iff there is EXACTLY one set, containing at least TWO relays.
- (e) iff there are exactly TWO sets, containing exactly TWO relays each.

The following are NOT COMPULSORY but why not try them?

- (f) iff there are exactly TWO sets, one with ONE relay and the other with TWO.
 - (g) iff there is exactly ONE set of THREE relays, but there may be other sets of any other size.
 - (h) iff there is exactly ONE set of two UNOPERATED relays! Watch this one!!
 - (i) if there are any number of sets of any size, EXCEPT that output must be cut off iff there is among them exactly ONE set of ONE or TWO relays.
- HINT : Design for a circuit which WILL give an output for one set of either one or two relays, and then graphically complement it.
- (j) iff there are exactly TWO sets, one of TWO relays and the other of any other size.

Hope you enjoy these! Just to ease your load, I'll set up camp and cook a meal while you folks get scribbling on your pads!

... End of Mile 17, sitting at marker "Mile 18"

+++

FOR THOSE WHO NEED TO KNOW

**68 MICRO
JOURNAL™**

FORTH

A Tutorial Series

By: R. D. Lurie
9 Linda Street
Leominster, MA 01453

FORTH09: A FORTH for OS-9 on the 6809

Here it is at last! Dan Johnson has come through for us just like the proverbial cavalry at the end of a John Wayne movie. He has written a completely full-featured FORTH-83 for OS-9. I hope that you will pardon me while I drool as I type this, since I am most excited with Dan's opus.

This product is called "FORTH09", and it has absolutely no relationship to the public domain minimal program of the same name which I discussed in my August, 1988, column.

General Description

FORTH09 is a greatly expanded 83-FORTH. It meets all of the requirements of the standard, plus it has a number of its own expanded features. In fact, FORTH09 has so many features that I cannot possibly do all of them justice in a short review. As one should expect from a quality product, FORTH09 includes an assembler and a screen editor.

The assembler covers all of the 6809 addressing modes, and, of course, it uses RPN, so beginners will think that the statements look a little confusing. However, it takes practically no time to become used to that. The relative branches are handled in a high-level format. This way, the programmer does not have to worry about whether a branch is short or long. The manual contains an entirely adequate explanation of how this works.

The screen editor comes configured for 3 standard terminals and for the CoCo3. Just choose the type that you are using and compile the proper screens. If your terminal doesn't fit one of the current choices, you can find instructions on making the changes to one of the existing forms or on writing your own.

There is a kernel of code which must be loaded at startup, but the rest of the code is in screens which can be compiled as needed. Therefore, one can tailor a version of FORTH09 to a specific job without carrying a lot of excess code which does nothing but occupy RAM or ROM. The manual says that by choosing to do so, efficient ROM code can be generated (I have no way to burn a ROM, so I can't check this, but I don't doubt it!)

The important OS-9 calls are available directly from FORTH09, so this would eliminate a lot of the need for assembly language routines. In fact, I am such a klutz with OS-9 that I was amazed when I could use the calls so easily. Certainly, I had a lot less trouble than I had with C or BASIC09! Furthermore, I hope to use FORTH09 to teach myself more about how OS-9 operates. Talk about "instant gratification"!!!

Scope

FORTH09 is the epitome of wide usefulness. The same version works on both Level I and Level II OS-9. The only reason I can think of for having two working copies of the system disk is that you do need a different boot for Level I and Level II. Otherwise, it would be difficult to know which version you were using. As I see it now, I think that FORTH09/Level I would be most useful for machine control and FORTH09/Level II would be best everywhere else.

As I mentioned before, FORTH09 has been designed to work well on virtually any 6809 machine running OS-9. I only have OS-9 for the CoCo, so I can't check this for sure, but it looks fine to me from reading the manual and the screens. Other than disk format, the only differences that I anticipate would arise from the peculiarities of a particular terminal configuration.

I will confess that after I received my review copy of FORTH09, I finally realized just how much I HATED OS-9! Well, that has suddenly changed. I can't say that OS-9 is my favorite operating system, yet, but FORTH09 has certainly turned the corner for me.

Detailed Description

FORTH09 differs from the usual FIG model in several important ways. First of all, this is a subroutine-threaded code, rather than the string of execution addresses which are indirectly loaded and executed by NEXT. Whenever speed is important, the compiler can be signaled to compile a definition in line, rather than as a subroutine jump. This is not routinely done, because it can make a significant difference in the length of the code, but it is a dynamically available option.

Two separate dictionaries are maintained by FORTH09. They are called the PRIMARY and the SECONDARY dictionary. The significance is that it is possible to SAVE only the PRIMARY dictionary, thus creating a minimum sized block of code which can stand alone, as in ROM. However, the SAVESYS command saves both dictionaries, which is important since the assembler and editor are entirely in the SECONDARY dictionary. Following Dan's recommendation, I normally select the SECONDARY dictionary for all of the definitions which I write and then do a SAVESYS; after all, disks are cheap!

Each dictionary header is separate from its code. I don't know what difference that would make to the average user, but it would make it virtually impossible for somebody to reverse-engineer a ROM made with FORTH09.

Dan has provided a large number of additional definitions beyond 83-FORTH. This includes a pretty complete line of double-number words, so there would probably be no need for a floating-point package; 32-bit numbers generally exceed the capacity of most floating-point packages, and scaling should be used for the big number/little number problem, anyway.

Dan has exerted a little bit of author's license in changing some of the familiar FORTH words to fit in with his ideas. As examples, he has DDUP instead of 2DUP and ENDIF instead of THEN as his preferred forms. However, he has provided the common synonyms, also, so this does not present a problem. In fact, he has provided the word EQU which generates a synonym for any existing word, so you can customize your own package, if you want to. All of these changes are adequately covered in the glossary and will never cause you a problem, unless you are working with the minimum possible memory.

Caveats

A couple of points should be made about the differences between FORTH09 and some other FORTHS, which might otherwise fall between the cracks. All of the problems mentioned here are related to transportability of programs and data between various FORTHS. If you are not concerned about that, then you can skip the next few paragraphs.

Decimal point within a number: It is usually acceptable to enter a number containing a decimal point, even though that decimal point will be ignored during integer math operations. However, the location of the decimal point is saved in a variable, usually called DPL, for use in some later operations. I have not been able to make FORTH09 accept a number containing a decimal, no matter how I have tried. Furthermore, DPL is not provided, so you will have to use your own. As I write this, I don't know how to "fix" it, but I will ask Dan to look into the matter. He has said that he intends to release a floating-point package, which should eliminate the whole problem, but it can interact unfavorably with existing programs which use math operations.

Flagging double numbers: All other FORTHS flag the keyboard input of a double number by terminating the entry with a ".", but FORTH09 does this by preceeding the first digit with a "#". I can't see that this would affect the portability of FORTH code, since the only change is during keyboard entry, and not during program execution. The only problem might come from reading data from a text file on disk, but even that appears to be a little far-fetched. All in all, I think that the only real problem could come from the frustration of having to learn another way to enter keyboard data, but even I learned, so you can, too.

Screen #1

```
\ Test samples                                \ RDL092188 |0
                                                |1
SECONDARY                                       |2
                                                |3
: EX0  ( - )                                   |4
  CR ." This is the first example." CR ;      |5
                                                |6
: EX1  ( - )                                   |7
  CR ." This is the second example." CR ;     |8
                                                |9
: EX2  ( - )                                   |10
  CR ." This is the third example." CR ;      |11
                                                |12
: EX3  ( - )                                   |13
  CR ." This is the fourth example." CR ;     |14
                                                |15
```

Screen #2

```
\ The first experiment                                RDL092188 |0
                                                    |1
: TABLE1                                             |2
  CREATE                                             |3
  DOES>      SWAP 2* + @ EXECUTE ;                  |4
                                                    |5
TABLE1  EXPERIMENT1                                \ create the vector table |6
      ] EX0 EX1 EX2 EX3 [      \ load the table      |7
```

Vector table generation: There are two common ways to generate vector jump tables; one way is shown in Screen #2 and the other in Screen #3. At this point, I simply want to say that the method shown in Screen #2 will crash in FORTH09, but the method in Screen #3 works fine. This appears to be a function of the way FORTH09 is built, and is not something that can be changed. On the other hand, I think that the method shown in Screen #2 is very poor programming practice and should be killed, any way. However, many programs exist which use the scheme of Screen #2, so watch out for it!

Shadow screens: FORTH09 requires that you use shadow screens. Unless you can think of some trick to play on the system, these screens are only useful for documenting the corresponding "regular" screen. There is no real harm, here, except that I don't like to be forced into an unfamiliar documentation scheme. I much prefer to document directly on the program screen so that I can see the comment and the code at the same time. This is only possible with shadow screens when you have a 132-column display. Also, I admit to a rather silly prejudice against "wasting" so much disk space.

SOURCE: D. P. Johnson, 7655
Southwest Cedarcrest St., Portland, OR
97223. \$150 (plus \$3 S/H).

VECTORED JUMP TABLES

In my last column, I discussed forward addressing and showed a rather elegant way to generate a vectored jump table. I would now like to discuss another way which used the CREATE ... DOES> construct; it is not as elegant, but is much simpler to type. However, it will not do the job of cross-selecting menus which was discussed last time; try it and you will see why.

Screen #1 simply defines the four words to be selected by our jump table. These definitions must be written before the table is defined, but can be written after the defining words TABLE1 and TABLE2. I have placed them in the first screen simply for convenience in illustrating the point.

Screen #2 uses CREATE ... DOES> to define a "defining word" called TABLE1. Notice that CREATE simply leaves an open space to be filled by the vectors at a later date. No specific amount of space is reserved at this time, so we have not specified how large the table is to be. The DOES> specifies what is to be done at run-time with the data in the table, but not how the data are to get there.

Line #6 of Screen #2 actually creates the vector table. Notice that this line is actually a command line which is executed when the screen is first loaded, and it is not a part of a definition to be executed later. The format of the command is simply its name TABLE1, which invokes its name, and EXPERIMENT1, which is the name of the jump table. Nothing could be simpler, which probably accounts for the popularity of this format.

All that is necessary to load the table named EXPERIMENT1 is the command shown in line #7. This must immediately follow the table invocation, or the scheme won't work. This line works because] immediately turns on the compiler and loads into the dictionary the execution addresses of the following words. The [immediately switches back to the interpreter, thereby stopping the addition of execution addresses to the dictionary.

Screen #3 starts off pretty much the same way, but CREATE takes a number from the Data Stack and allocates the actual storage space for the table during compilation. This way, the size of the table is set explicitly and is fixed as soon as TABLE2 is invoked. Therefore, it is not necessary to fill the table immediately, but it can be done at any time. This is a great programming convenience.

The action of DOES> in this screen is exactly the same as in Screen #2. You can see this with a Data Stack diagram, which you should construct if you need more help understanding the following explanation. The definition is entered with the address of the first element of the table as the top of the stack and the offset into the table as the next element of the stack. Therefore, the SWAP is necessary to get the offset into the right place. 2* multiplies the offset by the 2 bytes taken by a 16-bit address and + adds the result to the starting address. @ calls the execution address and EXECUTE jumps to it.

Screen #3

```

\ The second experiment                                RDL092188 |0
                                                         |1
: TABLE2                                              |2
  CREATE 2* ALLOT                                       |3
  DOES> SWAP 2* + @ EXECUTE ;                          |4
                                                         |5
4 TABLE2 EXPERIMENT2 \ create the vector table       |6
                                                         |7
\ EX0 \ EXPERIMENT2 >BODY 0 2* + ! \ load the table   |8
\ EX1 \ EXPERIMENT2 >BODY 1 2* + !                   |9
\ EX2 \ EXPERIMENT2 >BODY 2 2* + !                   |10
\ EX3 \ EXPERIMENT2 >BODY 3 2* + !                   |11

```

The vector table is created by the command in line #6. The first command must be a number which sets the size of the table, 4 in this case. This means that there will be 4 addresses stored in a table named EXPERIMENT2. Again, this is pretty much the same as in Screen #2, with the exception of the size of the table being on the Data Stack before calling TABLE2.

The remainder of Screen #3 simply shows one way to fill the vectored jump table named EXPERIMENT2. Notice that I have used an expanded format for each of the four commands required to load the four execution addresses into the table called EXPERIMENT2. This makes the commands self-documenting, which is a big

help. It shows which address goes where in the table. It also makes it easy to change the table later, but this smacks too much of self-modifying code to be to my taste. It is much more difficult to modify TABLE1, because it is harder to find which address went where (this may be an advantage!?!).

Either scheme, Screen #2 or Screen #3, will work with most versions of FORTH, but not with FORTH09. You can only use the procedure in Screen #3 with FORTH09. However, the algorithm implemented in Screen #3 is better for several reasons, so I recommend it, no matter which FORTH you are using.

CORRECTION

A line got left out of the definition of DO-IT in the last column. The first line of the definition should be

R> DROP

followed by the rest of the definition. Without this addition, the Return Stack will eventually overflow. I did not catch the error simply because I did not cycle through the example enough times. Dave Angel caught this for me when I showed him the code. This just proves that, no matter how hard I try and how hard I test, I cannot guarantee that my code will always be perfect!?!.

+++

FOR THOSE WHO NEED TO KNOW

68 MICRO
JOURNAL™



The Macintosh™ Section

Reserved as

A place for your thoughts

And ours.....

Mac-Watch

A Review of ThePerfectWORD™ Software for Bible Study and Research

*By James E. Law
1806 Rock Bluff Rd.
Hixson TN 37343*

If I said that the Macintosh can be all things to all people, you would no doubt think the case to be somewhat overstated. Even Apple's Mac evangelists wouldn't go quite that far. It is true, however, that the Macintosh (and other computers) can serve an almost unbelievably wide variety of needs from spreadsheets for Accountants to data bases for a Zinc mining company. It should be no surprise then, to note that software is now available to support the serious Bible student and researcher.

Many people believe that the Bible is the inspired word of God, and that it tells men how to live in order to be saved. That makes it worthy of careful study. Fully understanding its message, however, takes considerable diligence and calls for the use of every tool at one's disposal. The apostle Peter said that Paul wrote "some things hard to be understood." Most Bible students won't have any difficulty in agreeing that the Bible is a challenging book.

Star Software, Inc. (229 Live Oaks Blvd. Caselberry, Florida 32707) has developed several software packages to assist students and researchers in studying the Bible. One of these, ThePerfectWORD, is a fast and powerful search engine for finding, displaying, saving, and printing verses meeting specific search criteria. I have used this program frequently over the last two months and in this review, I'll share my experiences with you.

Getting Started

When ThePerfectWORD is opened, a display window and an entry window appear. You enter search criteria in the entry window and the verses identified in the search will be shown in the display window. The entry window can be hidden when not in use to provide more room for text. The text can be displayed in 9, 10, or 12 point type, and the spacing between verses can be adjusted.

Show Me the Verse

One of the simpler uses of ThePerfectWORD is to view selections of text. Select the "Show" mode, enter "James 2:24," and instantly this verse appears on the screen. This portion of the program is extremely flexible and allows precise selection of just the text you want to see. For example, you might want to see Romans 4:1-25; Hebrews 11:1; 11:8-19; and James 2:14-26 at the same time. No problem. Just enter the desired verses in an easy to understand format and they all appear. Considerable abbreviating of book names is allowed. The window has standard zoom button, scroll bars, and a grow box to view text when the selection is more than can be seen at one time.

Where Does It Say...?

By far the most important capability of ThePerfectWORD is its ability to find verses which satisfy prescribed search criteria. The feature is perfect when you want to know where the verse is that says that Jesus is a "propitiation" (just enter the word "propitiation" in the entry window.) A more common use might be to view all the passages that mention a specific word (e.g., "marriage"). You are not limited to searching on words, but can also search on phrases (e.g., "In Christ").

ThePerfectWORD searches are FAST! For example, a search of the entire Bible for the word "covering" resulted in 43 verses being found in about 6 seconds. A search of the entire Bible for

verses that contain both the word "faith" and the word "grace" took on the order of 3 seconds. The time required for the search seems more related to the number of "hits" than the scope of material being searched. Several months ago, I reviewed a competing Bible study program, and at the time was favorably impressed with its speed. Little did I know! It quickly became apparent that ThePerfectWORD was an order of magnitude faster than its most advertised competitor.

ThePerfectWORD allows you to set the limits of your search. You may enter specific starting and stopping points. A quicker method for setting search limits is to check *Quick Set* dialog boxes such as Old Testament, Poetry, Entire Bible, Gospels, or Epistles. There is a block for almost all searches that I make, so detailed entries were rarely needed.

The PerfectWORD provides great flexibility in specifying search criteria. A string of different words or phrases may be entered and connected with symbols for "or", "and", or "not." For example, by entering "faith, grace" you will be shown all verses which contain both the word "faith" and the word "grace." Similarly, the entry of "Adam, Eve" will display all verses which mention "Adam" but do not mention "Eve." Parentheses may be used for nesting in more complex searches. This provides great power to the researcher in indicating the subjects to be identified in the search. Searches involving complex logical strings are, of course, much slower than simple word searches. Note that the logic statements are not necessarily limited to words within a given verse. For example, if the same sentence used both the terms "grace" and "faith" yet the words fell in adjacent verses, these verses could still be identified in the search example cited above. ThePerfectWORD allows the context range for input logic statements to be specified. For example, the user can say, "I want to see the places where "grace" and "faith" are mentioned within 3 verses of each other (or 1 verse, or 2 verses, etc.).

The result of searches may be displayed in two formats. Either the references only may be entered (i.e., the book, chapter, and verse) or the references with text may be displayed. An serious limitation of ThePerfectWord Revision 1.0 was that only 100 verses with text could be viewed. Fortunately, this limitation was removed in Revision 2.0 by allowing the full text of searches to be displayed regardless of the number of verses involved.

Let's See it in Context

Often, it will be useful to see the verses satisfying your search criteria in context, that is, with some preceding and following verses displayed. With other Bible search programs, you will have to pull out your Bible and look each verse up. Not so with ThePerfectWORD. Simply double click any verse and it is instantly displayed within a separate window in context. You can have ThePerfectWORD display from 1 to 10 verses before and after the verse of interest. This is a most useful feature since the division of the Bible into verses often causes breaks in the middle of sentences and ongoing thoughts.

Word Counts

Have you ever wondered how many times a particular word is used in the Bible? If so, your curiosity can be satisfied. In seconds, ThePerfectWORD can count all the occurrences of a word in the designated range. For example "Jesus" is used 983 times in the Bible, and Paul uses the word "law" 78 times in Romans. ThePerfectWORD can search for all words except about 50 very common words like "and", "the", and "a."

You can count a string of words at the same time. Enter each word you want counted separated by a space, comma, or semicolon, and the word count for each will be displayed.

ThePerfectWORD can even count words based on entry of an abbreviation. For example, if you wanted to count all forms of the word "believe", you could enter "believe" and the words "believe, believes, belief, believeth, believest, believing, and believed" would be counted and the totals for each displayed. The ability to use abbreviations to widen a count is also useful in accurately scoping out a search for verses meeting the search criteria. In the above example, a search for "believe" would result in verses containing any of the above words being displayed.

Side by Side Comparisons

One of the most impressive ThePerfectWORD features is its ability to open numerous windows at once. For example, you can display Joel 2 in one window, open a new window for Daniel 2, and then display Acts 2 to see passages on prophecy and prophecy fulfilled. Similarly, you could display at the same time, passages from each of the four Gospels dealing with a particular parable. You can do the same thing with the results of word searches. For example, in one window you could display all passages from Romans with the word "faith" and in another window all passages mentioning "works."

Window management is easy. The *Windows* menu lists all currently open windows and allows

you to activate any of them. If you select *Tile Windows* from the *Display* menu, all text windows are neatly arranged side by side. With this feature you can easily view up to 5 windows at once. If you select "stack windows," the windows are placed in a staggered fashion behind one another so that while only one window is fully exposed, the titles of all windows are displayed. Regardless of the option chosen, any window clicked becomes active and may be scrolled to see more text.

Saving Your Work

It's likely that often you will want to save the results of your research. ThePerfectWORD provides several possibilities. First, you may export the contents of windows through the clipboard. Second, you may save any window as a text file which can be read by your word processor or page layout program. Finally, you may create a verse file.

The verse file option saves the results of your research as a list of verses without the accompanying text. This makes such files extremely compact. When you open such a file, you will have the option of viewing only the references (i.e., the book, chapter, and verse) or also see the text. The *Open*, *New*, *Save*, and *Save As* commands work with Verse Files just like other files.

Using the appropriate menu selections, you can add or delete verses to a verse file and can combine multiple verse files together. You can select the *Sort and Merge* option to remove duplicate verses and put the remaining ones in Bible order. You may also add a descriptive header to each verse file. This description will be saved, displayed, and printed with the verse file.

Printing Options

ThePerfectWORD window may be printed by selecting *Print* from the *File* menu. You may choose font size (9, 10, or 12) and verse spacing, but may not specify font style. If you want to alter the format significantly, you will need to transfer the information via a text file or the clipboard to some other application.

General Information

ThePerfectWORD can be used with the Macintosh 512E, Plus, SE, and II. Supposedly, you can operate with only one disk drive if you have at least a megabyte of RAM, but I suspect you would need the patience of Job to do so. Any program whose files occupy 2.6 megabytes calls for a hard disk for

efficient operations

ThePerfectWORD is such a simple program that you will rarely need to refer to the manual. When you do, you will generally find it to be adequate. I did have some problems understanding how to use the verse file feature, but in retrospect, this was probably a result of my failure to read carefully enough and not that the manual was unclear.

Other Related Products

The text modules used in this review were the King James Version and the New International Version. Each English language translation is \$75. Star Software also offers a Greek translation for \$150 and a Hebrew translation for \$180 translations. These enable Bible researchers to perform searches on the original Greek or Hebrew word.. The Hebrew version is entered and displayed from right to left on the screen since this language is read and written in this manner. These nonEnglish translations were examined only briefly during this review since my only language is Southern English. These products appeared to perform as advertised.

Do You Need ThePerfectWord?

I have used ThePerfectWORD frequently over the last two months and find it to be an exceptional program. Serious Bible students or researchers with access to a Macintosh cannot afford to be without this program. It does its thing so well that concordance books are made hopelessly obsolete. The speed of ThePerfectWORD is truly awesome. I had a number of concerns about limitations in Revision 1 of ThePerfectWORD but Star Software corrected all of them in Revision 2 leaving me little to gripe about. Users of Revision 1 should contact Star Software about upgrading to the new version since it is so much improved.

As a Macintosh software reviewer, I have access to a fairly wide variety of software. Most of it, however, gathers dust after the review. I can assure you that such will not be the case with ThePerfectWORD.

ThePerfectWORD is a quality product which deserves your attention. I recommend it.^{EOF}

FOR THOSE WHO NEED TO KNOW

**68 MICRO
JOURNAL™**

Intelligent Write / Erase Cycle Stress of MC68HC11 EEPROM Devices

Robert J. Pinteric
Motorola, Inc.
6501 WM Cannon Drive West - OE312
Austin, TX 78735
(512) 440-2077

Devices with EEPROM memories are tested in several different ways to screen units which would fail prematurely in an end-user's system. These tests typically include functional, pattern, data retention, and write / erase (W/E) cycling endurance testing. The former of these tests can be performed in a relatively short period of time using a high speed MOS test system, resulting in a low test cost per unit. However, the repeated W/E cycle stress of an EEPROM device takes a significant amount of time to complete. Screening units in this fashion using a high speed test system would result in high test costs per unit and low test throughput.

In order to minimize test costs and increase test throughput of Motorola MC68HC11 EEPROM Microcontroller units (MCU), a system has been developed which can simultaneously perform W/E cycling on up to 50 devices. These systems utilize the on-chip subsystems of the MC68HC11 in a unique way which enables each device to literally test itself. The system software enables W/E cycling of five different MC68HC11 MCU devices: MC68HC11A8, MC68HC811A2, MC68HC811E2, MC68HC11E9, and the MC68HC11F1. Although these systems are primarily used to perform production W/E cycling, it is also used to perform engineering evaluations. Furthermore, the system can be software configured to function as an MC68HC11 EEPROM gang programmer.

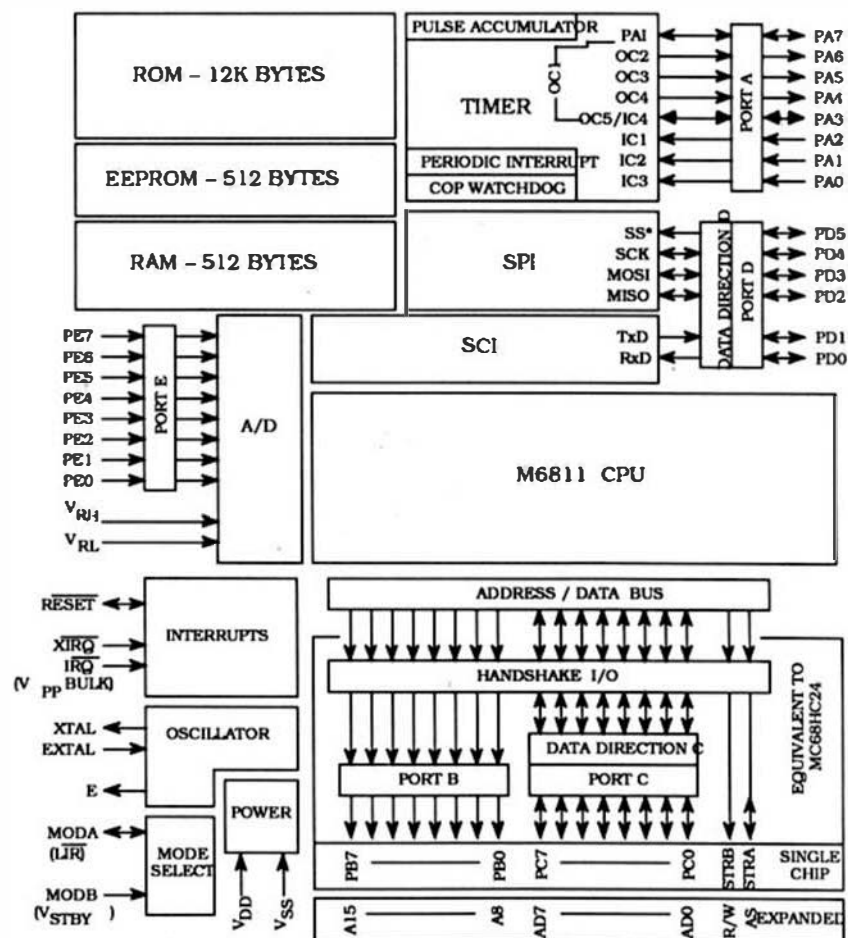


FIGURE 1 MC68HC11E9FN BLOCK DIAGRAM

DEVICE NUMBER	ROM	RAM	EEPROM	COMMENTS
MC68HC11A8	8K	256	512	ORIGINAL 68HC11
MC68HC11E9		12K	512	MORE ROM & RAM
MC68HC811A2	NONE	256	2K	NO ROM
MC68HC811E2	NONE	256	2K	4 INPUT CAPTURES
MC68HC11F1		NONE	1K	512 NON-MUX BUS

TABLE 1 MC68HC11 FAMILY OF DEVICES.

MC68HC11 Overview

In order to fully comprehend the function of the W/E cycle system, a brief overview of the MC68HC11 is necessary. As mentioned previously, there are five different members of the M68HC11 Family, each with different memory sizes and features, as listed in Table 1. For simplicity, the MC68HC11E9 MCU will be discussed. The MC68HC11E9 MCU, shown in Figure 1, contains 12k bytes of mask programmable ROM, 512 bytes of RAM, and 512 bytes of EEPROM. In addition to memory, the MC68HC11 contains an 8 bit 8 channel A/D converter, two serial ports, a 16 bit free running timer with 3/4 input capture lines and 5/4 output compare lines, a real time interrupt, and a pulse accumulator. The MCU utilizes an enhanced M6801 instruction set with 88 additional opcodes, including divide and bit manipulation instructions.

The EEPROM memory is enabled when the EEON bit in the CONFIG register is set. The write (or programming) mechanism for the EEPROM is controlled by user programmable bits in the PPROG register. The erased state of the EEPROM is \$FF, thus programming changes a bit from a 1 to a 0. Programming and erasure of the EEPROM relies on an internal high-voltage charge pump.

MC68HC11 devices operate in one of four modes: 1) single-chip, where the device acts as a monolithic MCU, 2) expanded multiplexed, where two I/O ports become a multiplexed address/data bus, 3) special test, for factory testing only, and 4) bootstrap mode. The MC68HC11 is configured to run in the bootstrap mode when operating in the W/E cycle system. When configured in the bootstrap mode, the MC68HC11 is capable of receiving a serial stream of data via its serial communications interface (SCI) and storing the data to RAM. Once the serial transmission is received, the M68HC11 vectors to the beginning of RAM and executes the code.

Write / Erase System Hardware

The W/E system consists of two boards, a device under test (DUT) board, shown in Figure 2, and a driver board, depicted in Figure 3. The driver board controls the W/E stress of the M68HC11's, which are placed on the DUT board. The two boards interface via an edge connector. In production, W/E cycling occurs at 125 C. Thus, the DUT board is placed inside a high temperature oven and connects to the driver board located outside the oven through the edge connector which is placed inside the oven wall.

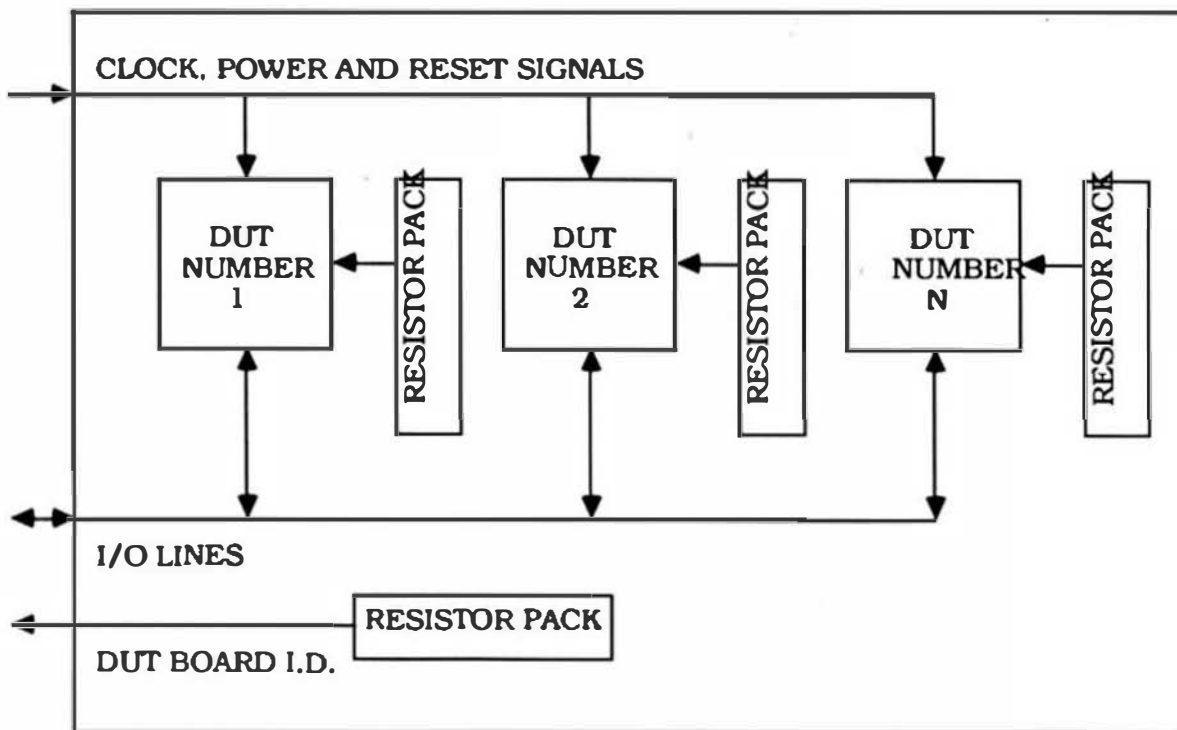


FIGURE 2 W/E SYSTEM DEVICE UNDER TEST BOARD.

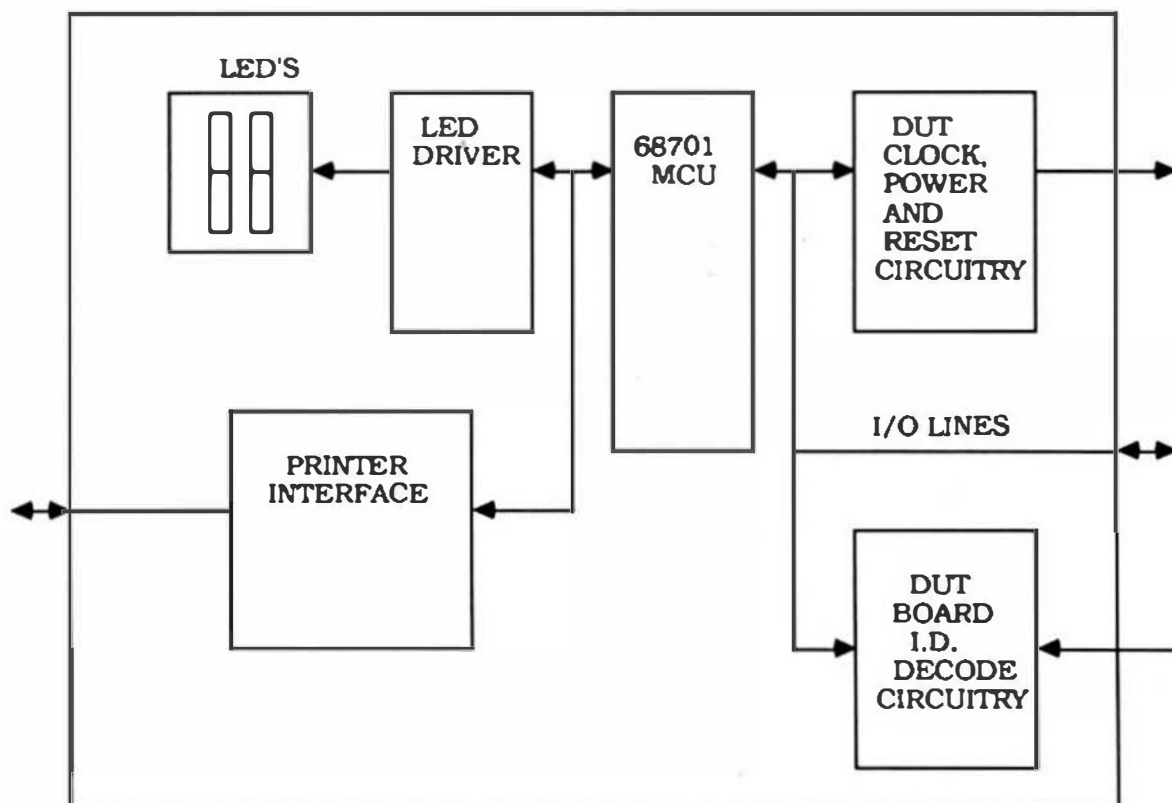


FIGURE 3 W/E SYSTEM DRIVER BOARD BLOCK DIAGRAM.

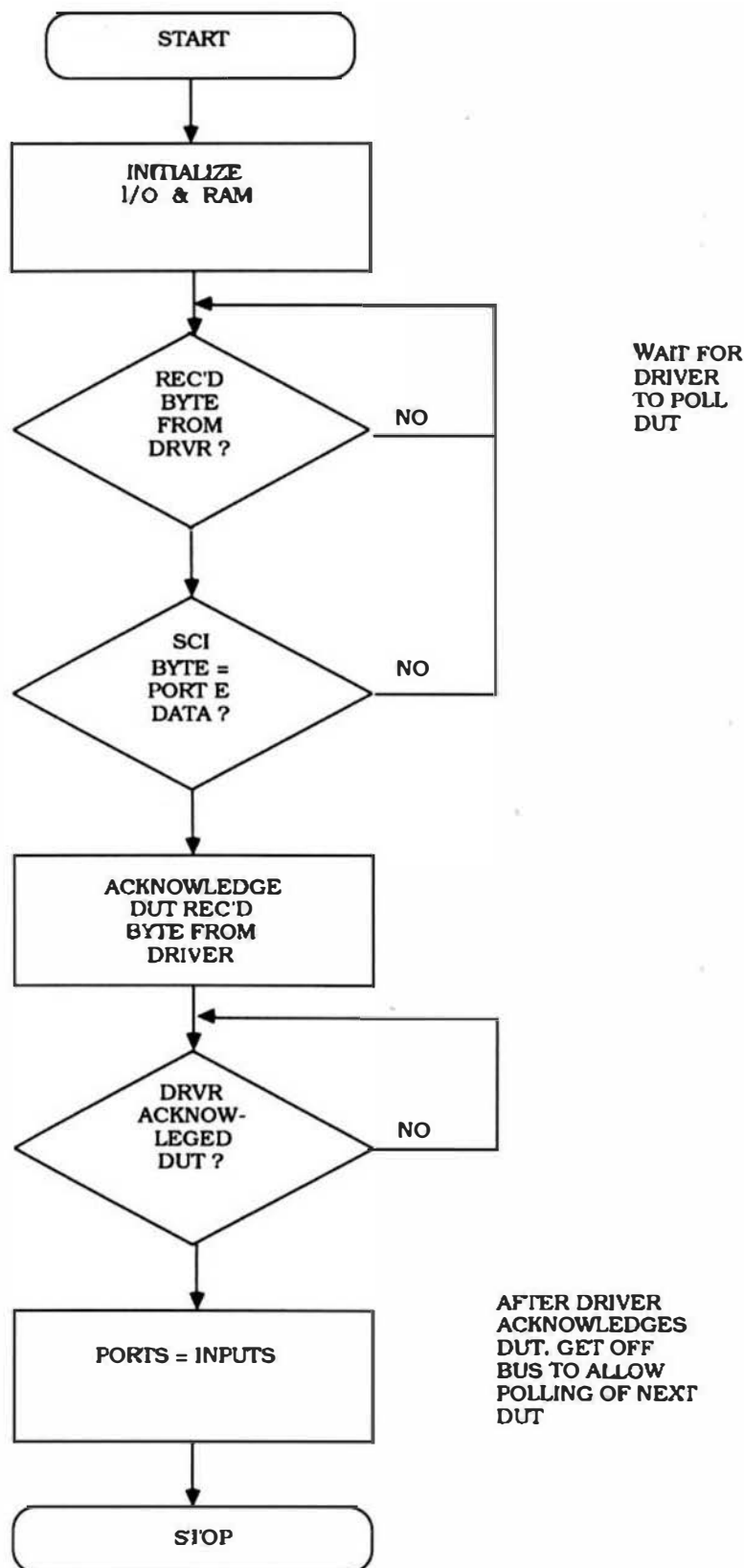
The MC68HC11 DUT board contains little circuitry outside of the MCU's themselves. Each MC68HC11 connects to an I/O bus in order to communicate with the driver board. Via the DUT socket, the port E pins of each MCU are hardwired with a hexadecimal address equivalent to the socket number in which it is placed. Since the I/O signals are bussed, this address is used by each DUT to discern whether the driver is initiating communication with itself or another

DUT. Clock, power and reset signals are also obtained from the driver board. The DUT's operate at an E clock frequency of 1.0 MHz with a 5.5 volt supply. Lastly, the DUT board contains a hardwired board identification code to inform the driver board of the package type and quantity of MC68HC11's on the DUT board. DUT boards were made to support 48 pin DIP, and 52 and 68 pin PLCC packages. (Note: see MC68HC11 Data Sheets for package information on particular MC68HC11 MCU devices.) Table 2 lists the quantity and the package type of the MCU's for the respective DUT board codes.

The driver board is primarily composed of a Motorola MC68701 MCU and several logic chips. The MC68701 MCU's features include 2k bytes of EPROM, 128 bytes of RAM, an SCI, and parallel I/O. The driver board consists of several sections which facilitates its use in both a production and engineering environment. As cited earlier, the driver board contains circuitry which provides the DUT's with power, reset and clock signals. Several I/O lines are present to communicate with the DUT's. Additional circuitry enables the driver to determine the DUT board identification code. This information is used by the driver board to configure its I/O circuitry to properly communicate with the DUT board. The LED's and printer interface sections of the driver board are used to

RESISTOR PACK NODE DUT PACKAGE TYPE			DUT QUANTITY	
N1	N2	N3		
0	0	0	PLCC	30
0	0	1	PLCC	50
0	1	0	DIP	15
0	1	1	DIP	30

TABLE 2 DUT BOARD IDENTIFICATION CODE.



interface with the user. Although a printer is not connected to the driver board during production W/E cycling, a push button switch contained in the printer interface is used to initiate W/E cycling. The printer is used extensively, however, in engineering evaluations as a data collection device.

System Operation and Software

The software program which runs the W/E cycle system consists of two parts: 1) M6801 driver board code and 2) M68HC11 DUT board code. Since the M68HC11 code is upwardly compatible with M6801 code, the two programs were combined into one larger program, which is assembled using an M68HC11 crossassembler. Care must be taken to assure that no M68HC11 specific opcodes are included in the M6801 portion of code, since the MC68701 MCU will view the opcodes as illegal instructions.

Once assembled, the software to operate both the driver and DUT boards is programmed into the driver board's MC68701 MCU. As mentioned previously, the DUT MC68HC11's operate in the bootstrap mode which enables them to use their SCI to read a program into on-chip RAM and execute the program. This special feature of the MC68HC11 bootstrap mode allows the driver board to download the MC68HC11 portion of the code to all the DUT's simultaneously. Programs downloaded to the DUT's are limited in size to 256 bytes in order to completely fit in the RAM area of all the MC68HC11's. Programs are serially downloaded from the driver board via the MC68701 MCU's SCI to the DUT board at a rate of 4800 baud.

Once the M68HC11's are loaded onto the DUT board and both the driver and DUT boards are placed in their respective positions about the high temperature oven, the system is ready for use. The driver board controls the W/E cycle stress of the MC68HC11's in the manner outlined in Figure 4. After initializing itself upon power up, the driver board

FIGURE 5 'QCHECK' PROGRAM FLOWCHART.

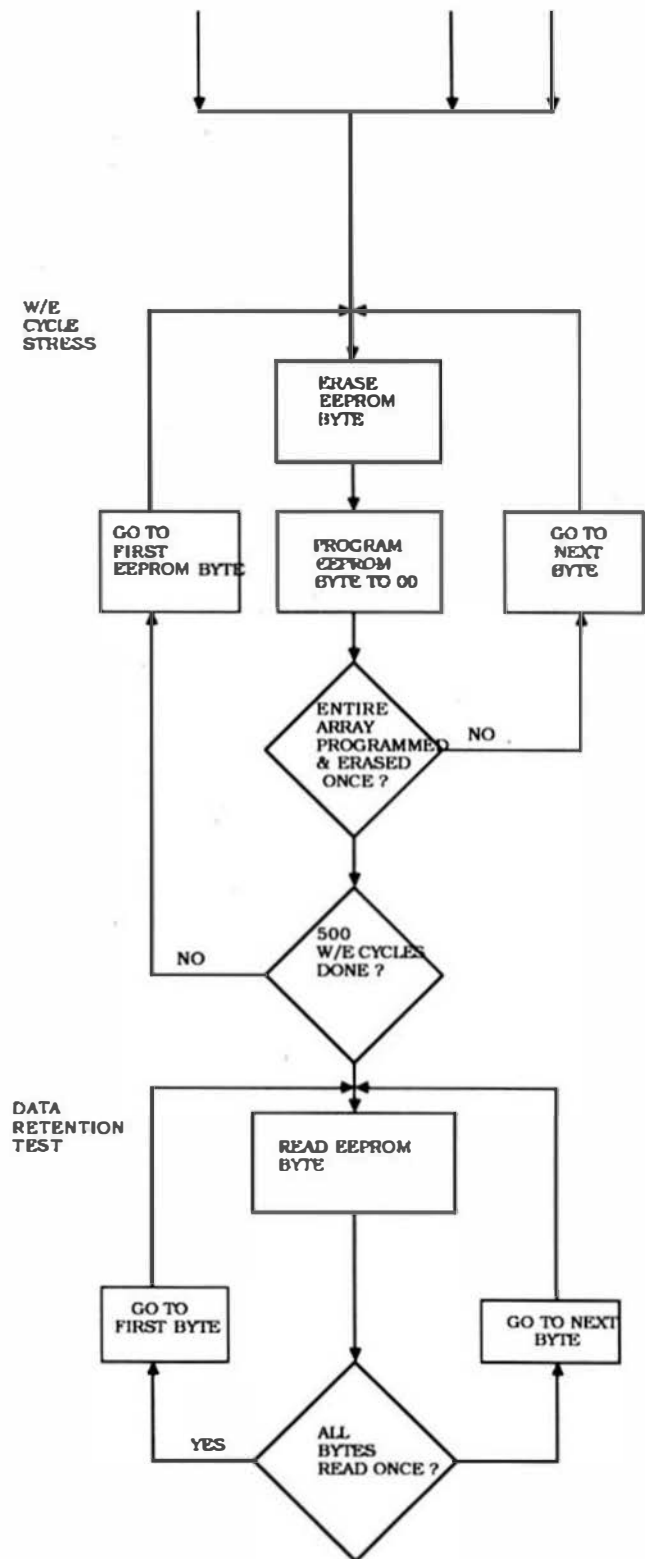
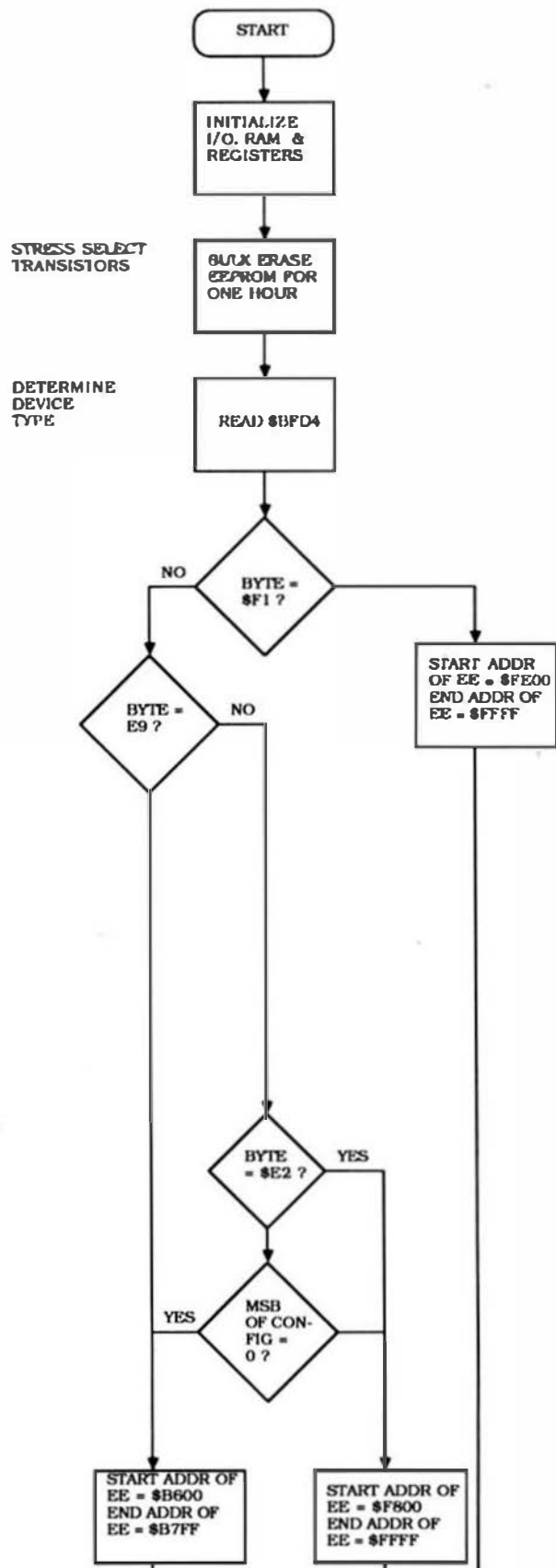


FIGURE 6 'STRESS' PROGRAM FLOWCHART

reads the DUT identification code to determine the type of DUT board in use. Recall, the driver board uses this information to properly configure buffers in order to communicate with the DUT board. If an invalid code is read, which would be the case if the boards were not properly connected, an error code would be displayed on the LED's.

Once properly configured for communication, the driver board downloads the 'QCHECK' MC68HC11 program to the DUT's. This program, outlined in Figure 5, is used to perform a quick socket check of the DUT's to assure that all the MCU's are properly seated within their sockets.

To perform a socket check, the driver board polls each DUT individually, and waits for the unit to respond. To poll a particular MC68HC11, the driver board transmits a data byte which corresponds to the socket number of the DUT. All of the DUT's simultaneously receive the data byte and compare it to the value hardwired to port E of their individual socket. When the data byte received matches the port E value, the MCU responds. After the proper handshake signals are exchanged, the MC68HC11 clears the signals applied to the I/O bus enabling the driver board to poll the next device.

After polling all of the devices, the driver board displays the socket number(s) of the DUT('s) which did not respond. This allows the user to power down the system and reinsert any device which is not correctly positioned. If all the DUT's respond, the display will flash 00. The driver board will continue to display the socket numbers which did not respond until the user presses a button on the driver board to initiate W/E cycling. Before cycling is initiated, however, the oven in which the DUT boards are placed is heated to 125 C.

Once the appropriate button is pressed, the driver board writes a continuous 00 to the LED's to indicate that W/E cycling was initiated. A second program, 'STRESS', outlined in Figure 6, is then downloaded to the DUT's. From this point on the driver board does not communicate with the MC68HC11's but solely provides them with power and clock signals. Additionally, all inputs from the user interface are ignored. Thus, W/E cycling can only be aborted by resetting the system.

Each DUT individually receives the 'STRESS' program and performs W/E cycling on its on-chip EEPROM. After initialization, the 'STRESS' software performs a one hour bulk erase of the entire EEPROM array. The bulk erase stresses the select transistors which are activated to access individual EEPROM bytes.

The 'STRESS' program was written to run on all current MC68HC11's, each with various EEPROM sizes and address locations as cited earlier (Table 1). Thus, once the one hour bulk erase is completed, the DUT must determine the size and location of its EEPROM. This is achieved by examining a location in boot ROM or the CONFIG register. On the newer MC68HC11 devices (E2, E9 and F1) boot ROM location \$BFD4 contains data which identifies the particular device type. For example, an MC68HC11E9 device contains 'E9'. From this byte, the start and stop locations of EEPROM can be determined for these devices. The MC68HC11A8 and MC68HC811A2, on the otherhand, do not contain this boot ROM data byte. Thus, the CONFIG register is used to distinguish between these two devices. If the upper nibble of the CONFIG register is 0, then the device is an MC68HC11A8. A value other than zero indicates that the device is an MC68HC811A2.

Once the device type is determined, 500 W/E cycles are performed on each EEPROM byte in order to screen units which contain weak EEPROM floating gate cells. Once 500 W/E cycles are completed, a data retention test is performed. The data retention test takes the form of continuously reading each EEPROM byte in succession until the system is powered down. Upon completion of W/E cycling the EEPROM of each device should be fully programmed to 00.

Once W/E cycling is completed, the oven is cooled down and the DUT MC68HC11's are removed from the boards. The units then receive a full test on a high speed tester. During the post W/E cycle test, units which do not initially contain 00 in their entire EEPROM are binned out as bad devices.

Conclusions

The two board W/E cycle system is a cost effective efficient tool to screen MC68HC11 MCU EEPROM devices which may fail prematurely in an end user's system. This versatile system is used to stress five different MC68HC11 devices with one single software program.

As mentioned earlier, the W/E cycle system can also be used for engineering evaluations by altering the software programmed into the driver board MC68701 MCU and downloaded to the DUT's. Software changes enable the driver to collect data from the DUT's, and to print data via its user interface.

Additionally, the W/E system can be programmed to act as a gang programmer for MC68HC11 devices. Thus, up to 50 MCU's can be programmed simultaneously with one driver board and DUT board.

FOR THOSE WHO NEED TO KNOW

**68 MICRO
JOURNAL™**

Bit-Bucket



By: All of us

'Contribute Nothing - Expect Nothing', DMW '86



MOTOROLA INC.

Microprocessor Products Group
6501 Williams Canyon Drive West
Austin, Texas 78735-0000

ALTOS ANNOUNCES MULTIUSER SYSTEM BASED ON MOTOROLA'S 68030

Seventh Company in Last Month to Announce 030-Based Computer

LAS VEGAS, Nev., Nov. 14, 1988 — Motorola's Microprocessor Products Group (Austin, Texas) today announced that Altos Computer Systems (San Jose, Calif.) will incorporate Motorola's top-of-the-line 68030 (030) processor in its highest performance multiuser system. The new computer, the Altos 68X Series 030, is configured to use two 25 MHz 030 chips and supports 250 users. The announcement was made at Comdex, a computer industry trade show.

The 030's compatibility with all processors in Motorola's 68000 family allows for easy migration of software from Altos' 68020-based products to its new 030-based multiuser system. Currently the 030 powers more than 50 systems, including personal computers and workstations from Apple Computer, Hewlett-Packard, NeXT and Sony Microsystems. These systems add to Motorola's \$100 billion installed base of hardware and \$3 billion base of 32-bit software, the world's largest.

NEW JAPANESE SYSTEMS ADOPT MOTOROLA'S 68030

Fujitsu, Sharp and Sumitomo-Denko Expand Portfolio of 030 Users

AUSTIN, Texas, Oct. 19, 1988 — Motorola's Microprocessor Products Group today announced that three major Japanese companies will incorporate its 68030 (030) microprocessor as the central processor in systems that are scheduled for shipment by the year's end. Fujitsu (Tokyo), the largest manufacturer of mainframes and minicomputers in Japan, will have two of its next-generation minicomputers on the 030. Sharp (Osaka) and Sumitomo-Denko (Tokyo) will both use the 32-bit processor in high-end workstations.

Fujitsu's 030-based A-60 and A-80 are the latest in the company's A-series of minicomputers to be based on Motorola's 68000 family. Sharp's IX-6 Model 2 and Sumitomo-Denko's U-Station/E40, both using the 030, are top-of-the-line engineering workstations that perform advanced CAD/CAM operations. The workstations add to Sharp and Sumitomo-Denko's current 68000-based product lines.

The 030's compatibility with processors in the 68000 line provides an easy upgrade path for more than 400 companies using its predecessor, the 68020. Currently, more than 50 companies use the 030 to power their systems. Sony Microsystems, the leading manufacturer of workstations

Shermaz Daver
Cunningham Communication, Inc.
(408) 982-0400

Lisa Hadley
Microprocessor Products Group
(512) 440-3095

"Our 68000 family has always provided a growth path for companies," said Murray A. Goldman, senior vice president and general manager of Motorola's Microprocessor Products Group (Austin, Texas). "Our future 68040 will significantly increase the performance of systems while maintaining compatibility with current 68000-based machines."

The new Altos 68X Series 030 runs the Altos Pick operating system and is available in three configurations ranging in price from \$25,000 to \$45,000. All three configurations of the system will ship in the first quarter of 1989.

Founded in 1977, Altos designs, manufactures and markets 16- and 32-bit networked multiuser solutions for distribution in 70 countries. More than 108,000 Altos systems have been installed worldwide.

Motorola's \$2.2 billion Semiconductor Products Sector (Phoenix, Ariz.), which includes the Microprocessor Products Group (Austin, Texas), is a part of Motorola Inc. It is the largest and broadest supplier of semiconductors in North America, with a balanced portfolio of over 50,000 devices.

In Japan, incorporating the 030 in its high-end workstations. Numerous companies in the United States, including Apollo Computer, Apple Computer, Hewlett-Packard and NeXT, Inc., also offer systems based on the 030.

"The versatility of the 030 allows it to be used in a variety of applications from personal computers to high-end workstations and supercomputers," said Murray A. Goldman, senior vice president and general manager of Motorola's Microprocessor Products Group (Austin, Texas). "The power of the 030 coupled with its price and installed software base continue to make it the choice of major system companies."

The 030 is a highly integrated microprocessor with an internal parallel (Harvard-style) dual bus architecture, memory management unit, and separate 256-byte data and instruction caches on a single chip. All processors in the 68000 family are fully compatible with one another, allowing for easy migration of software. Currently, Motorola's 68000 family has a \$100 billion installed base of hardware and a \$4 billion 32-bit software base, the world's largest.

Motorola's \$2.2 billion Semiconductor Products Sector (Phoenix, Ariz.), which includes the Microprocessor Products Group (Austin, Texas), is a part of Motorola Inc. The company is the largest and broadest supplier of semiconductors in North America, with a balanced portfolio of more than 50,000 devices.

SONY INTRODUCES THREE WORKSTATION LINES BASED ON MOTOROLA'S 68030

AUSTIN, Texas, Oct. 31, 1988 — Motorola's Microprocessor Products Group today announced that Sony Microsystems, the leading manufacturer of workstations in Japan, will incorporate Motorola's 68030 (030) processor and 68442 (882) floating-point math coprocessor in three workstation lines. Sony joins a host of vendors including Apple Computer, NeXT and Fujitsu that have recently announced systems based on the top-of-the-line Motorola processor.

Sony's three workstation lines, the NEWS 1700, 1800 and 1900 Series, use a 25 MHz 030 as the central processing engine and a 25 MHz 882 for advanced numeric operations. The 1700 systems incorporate a single 030 processor and are targeted as desktop-aided design tasks. The NEWS 1800 Series increases its own performance by including a second 030 as an input-output processor. The high-end NEWS 1900 workstation also uses a second 030 for graphics networking and VME bus support. These Sony workstations provide a broad spectrum of cost-performance solutions for a range of markets including engineering, software development and network server applications.

"Sony is clearly a technology leader and we are proud that our 68000 family has contributed to its success," said Murray A. Goldman, senior vice president and general manager of Motorola's Microprocessor Products Group (Austin, Texas). "Our 68000 family will continue to deliver the performance needed for leadership in the workstation market."

Motorola's 68000 family consists of the 68000, 68010, 68020 and 68030. All members of the family are compatible with one another, allowing for easy software migration from Sony's 68020-based products to the new 030-based workstations. Currently, the 68000 family has established a \$100 billion installed base of hardware and a \$3 billion installed base of 32-bit software, the world's largest.

Sony Microsystems Company, headquartered in Palo Alto, Calif., is a wholly owned subsidiary of Sony Corporation of America, itself a division of the \$11.4 billion Sony Corporation, Tokyo. Sony Microsystems markets state-of-the-art computer industry hardware and software in the United States.

Motorola's \$2.2 billion Semiconductor Products Sector (Phoenix, Ariz.), which includes the Microprocessor Products Group (Austin, Texas), is a part of Motorola Inc. It is the largest and broadest supplier of semiconductors in North America, with a balanced portfolio of over 50,000 devices.

CERTIFIED SOFTWARE CORPORATION

P.O. BOX 70, RANDOLPH, VT 05060 USA TELEPHONE: 802-728-4062 FAX: 802-728-4126

**CERTIFIED SOFTWARE ANNOUNCES BBS FOR
OMEGASOFT PASCAL ON 68000 SYSTEMS**

RANDOLPH, Vermont, October 19, 1988 - Certified Software today announced the installation of a Bulletin Board System to be used for customer support of the following products:

P20K - 68000 series Pascal for OS9 and PDOS hosts.
PXK9 - 6809 Cross Pascal for 68000 host running OS9.
PCSK - 68000 Pascal for Atari ST host.

The BBS allows users to call in using a modem at 2400 (V.22 bis), 1200 (V.22), or 300 (Bell 103) baud. The terminal program (supplied by Certified Software) allows error free communications with the BBS to facilitate downloading and uploading of programs reliably.

Access is on a subscription basis, with charges based on usage and a monthly account maintenance fee of \$3. The minimum amount to open an account is \$80. It is estimated that this would last the average user a year or more. Users apply for an account by logging into the system the first time and sending payment information to the BBS sysop (CSC).

The system is designed primarily to supply users with the latest information on OmegaSoft Pascal including patches and work-arounds to known problems. There will also be free software available on the system, and users may exchange software they have written, and even purchase software from other users (\$50 maximum per sale) by having money from their account transferred to the seller's account by the sysop.

The terminal program required to access the system will be included in future updates of existing products, or may be obtained from your distributor. Copies may be purchased directly from Certified Software at the above address for \$10 with your prepaid subscription of \$80 or more. Subscriptions may be charged to your credit card (VISA and Mastercard), credit card customers receive a 5% bonus.

®
OmegaSoft Pascal

**MOTOROLA INC.****Semiconductor Products Sector**

For further information, contact:

MOTOROLA INC.

Angela Hatfield, Press Relations, (602) 994-6900
Dev Chakravarty, Technical Marketing, (602) 821-4424

PCPI INC.

Steven J. Leon, Technopolis Communications, (213) 670-5606
Eric W. Gaer, Director of Marketing, (619) 485-8411

**MOTOROLA, PCPI™ INTRODUCE NEW
GENERATION OF LASER PRINTER CHIPS**

Las Vegas, November 14, 1988 . . . Motorola Semiconductor Products Sector and Personal Computer Products Inc. (NASDAQ symbol: PCPI) today introduced the first of a new generation of application-specific integrated circuits (ASICs) that lowers the cost and improve the performance of laser printer controllers.

"The LPC1 and ALPC1 gate array circuits are the keystones of an integrated array of products and services for manufacturers of printer engines, vendors of laser printers and systems integrators," said Eric Gaer, director of marketing for PCPI.

"PCPI's OEM customers may now purchase or license all the elements needed to build and market any advanced laser printer product solution -- chips, the custom development of software and hardware, turnkey controller designs, complete printer systems, emulations, an extensive font library, and ImageScript™, PCPI's emulation of PostScript®.

In a related announcement, PCPI introduced two printer controllers that incorporate the LPC1 and ALPC1 chips.

The chips mold PCPI's expertise in systems and software design with Motorola's two micron double-layer metal and advanced one-micron triple-layer metal CMOS (Complementary Metal Oxide Semiconductor) semi-custom circuit technology.

The two companies announced in late August 1988 an agreement to jointly design, manufacture and market a new generation of ASIC chips "that promises developers the vehicle to provide users with less expensive printers that feature greater text-and-graphics capabilities," said Gaer.

Both the LPC1 and ALPC1 combine into a single integrated circuit a variety of printer operations that previously required several chips. This simplifies controller design and customization, enhances functionality, improves performance, lowers costs -- and frees board space for additional functions.

For example, the LPC1 integrates into a single chip the interface with the central processing unit, dynamic RAM control, and the direct memory access video interface. Additionally, the ALPC1 includes on-board hardware assist for raster operations.

The jointly-developed ALPC1 chip for laser-printer controllers represents the first commercially available application for Motorola's high-density CMOS array (HDC Series) family of one-micron triple-layer metal semi-custom circuits, introduced in November 1987.

The ALPC1 (Advanced Laser Printer Controller) chip incorporates one of the most advanced silicon technologies available to provide sub-nanosecond speed combined with extremely small chip dimensions. Features of Motorola's HDC Series include densities up to 105,000 gates in a channelless sea-of-cells architecture that provides over 75 percent typical utilization. A one-micron drawn-gate-length CMOS process having triple-layer metal routing and power distribution results in 260 picosecond typical gate delays and very small die size (226 mils square for 16K gates.) The circuits are packaged in economical PLCC and fine pitch QFP surface-mount configurations and PGAs for through-board applications.

L.J. Reed, vice president and general manager of Motorola's Application Specific Integrated Circuits Division, said "high utilization in a one-micron channelless array provides a combination of performance and economy ideally suited to building application specific controller circuits."

In addition to providing silicon technology matched to the laser printer market, Reed said "the Motorola-PCPI chips illustrate the use of specialized library cells to speed the design phase of creating customized print-engine controllers."

The Motorola-PCPI ASIC family is designed to enhance the performance and functionality of systems based on Motorola's line of MC680X0 microprocessors. They will support up to 40-page-per-minute printers and standard emulation of text and bit-mapped graphics, and they are compatible with the most commonly-used printer-engine interfaces.

In addition, the ALPC1 is designed to provide 68000-based printers with 68020 performance at the lower cost of the Motorola 68000 chip.

Motorola's Application Specific Integrated Circuits (ASIC) Division in Chandler, Arizona is dedicated to the development and production of semi-custom and structured custom circuits using gate array, self-based and functional-block techniques. The ASIC Division manufactures and markets integrated circuits worldwide for commercial and military end-use markets and develops, acquires and maintains advanced computer-aided design software dedicated to the development and design of application-specific devices.

Located in San Diego, PCPI is a pioneer developer, manufacturer and marketer of laser printers and intelligent, microprocessor-based board-level products designed for use with leading microcomputers and workstations.

THE LPC1:

A LASER PRINTER CONTROLLER INTERFACE CHIP

The LPC1 is the first of a new generation of Laser Printer Controller chips. It incorporates the three primary functions of a laser printer controller: the CPU interface, the Dynamic RAM control, and the video DMA interface. The LPC1 is fully compatible with Motorola's line of MC68010 microprocessors.

Contained within the LPC1 are eight programmable registers which provide a wide range of flexibility in customizing the LPC1 for a specific engine application. On-board CPU clock driver, programmable serial port clock, 16-bit interrupt timer, and a special CPU controlled Diagnostic printer emulation mode are just a few of the LPC1's unique features.

FEATURES

CPU INTERFACE

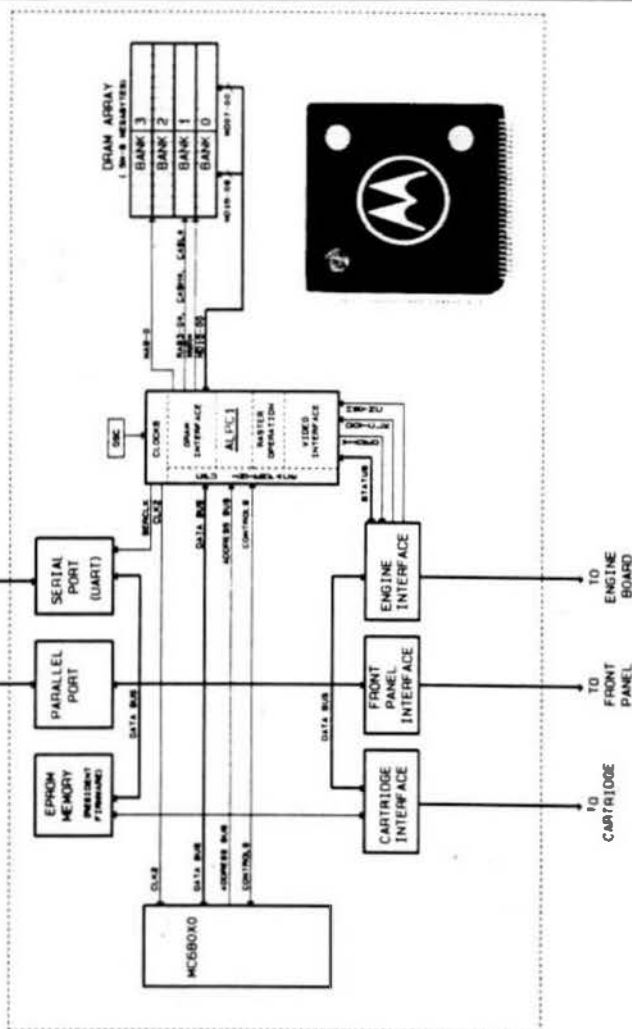
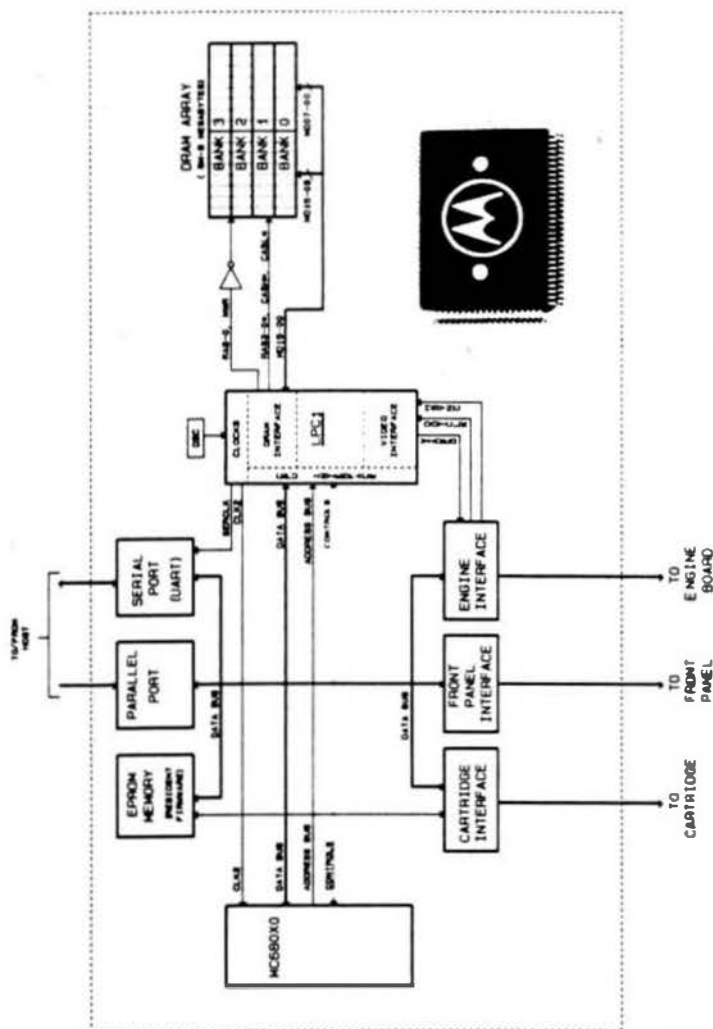
- MC68010 interface (10-12 MHz)
- CPU Clock Generator
- Variable DTACK Selection
- CPU Interrupt or Polling Option
- CPU Controlled Diagnostic Printer Emulation Mode
- 16-bit Programmable Software Interrupt Timer
- 16-bit Data Bus

DRAM CONTROL

- 0.5M - 8 Megabytes DRAM Controller
- 256K and 1Meg DRAM Chip Support
- 3-Port Memory Interface Arbitration (CPU, Video DMA, Refresh)
- Internal Refresh Cycle Control (RAS-Only)

VIDEO DMA INTERFACE

- Video DMA with selectable Write-back Zero Option
- 16-bit Programmable Vertical Line Scan Counter with Interrupt or Polling Option
- Internal/External LGATE Option
- Internal 12-bit Programmable LGATE Counter
- 12-bit Programmable Words/Line Counter
- 22-bit Programmable Video DMA Memory Address Counter with optional Auto Wrap on Scan Counter Interrupt



THE ALPC1:

AN ADVANCED LASER PRINTER CONTROLLER INTERFACE CHIP

The ALPC1 is the first of a new generation of Advanced Laser Printer Controller chips. It incorporates the three primary functions of a laser printer controller: the CPU interface, the Dynamic RAM control, and the video DMA interface. Additionally, the ALPC1 contains on-board hardware support for raster operations. The ALPC1 is fully compatible with Motorola's line of MC68010 microprocessors and is a successor of the LPC1's first generation LPC1 chip.

Contained within the ALPC1 are several programmable registers which provide a wide range of flexibility in customizing the ALPC1 for a specific engine application. On-board CPU clock driver, programmable serial port clock, 16-bit interrupt timer, engine refresh serial port, and a special CPU controlled Diagnostic printer emulation mode are just a few of the ALPC1's unique features.

FEATURES

CPU INTERFACE

- MC68010 interface (10-16 MHz sync & 16-25 MHz asynch operation)
- CPU Clock Generator
- Variable DTACK Selection
- CPU vectored Interrupt or Polling Option
- CPU Controlled Diagnostic Printer Emulation Mode
- 16-bit Programmable Software Interrupt Timer
- 16-bit Data Bus
- Zero wait state < 1 cycle 1 = 64 state Read cycle

RASTER OPERATIONS

- 3 Operand Raster Operations (source, mask, destination)
- Hardware Source Alignment to Destination
- Automatic Destination Address Generation
- Data Preservation (clipping, on Destination)

DRAM CONTROL

- 0.5M - 8 Megabytes DRAM Controller
- 256K and 1Meg DRAM Chip Support
- 3-Port Memory Interface Arbitration (CPU, Video DMA, Refresh)
- Internal Refresh Cycle Control (RAS-Only)

VIDEO ENGINE INTERFACE

- Video DMA with selectable Write-back Zero Option
- 16-bit Programmable Vertical Line Scan Counter with Interrupt or Polling Option
- Internal 12-bit Programmable LGATE Counter
- 12-bit Programmable Words/Line Counter
- 22-bit Programmable Video DMA Memory Address Counter with optional Auto Wrap on Scan Counter Interrupt
- Engine command status Serial Port Interface

Computer Systems Consultants, Inc.
1454 Latta Lane, Conyers, GA 30207
Telephone Number 404-483-4570/1717

CMODEM

CMODEM is a menu-driven telecommunications program designed to facilitate the transmission of data between microcomputer systems and terminals or other remote computer systems. It functions in several modes of operation.

In the simplest mode of operation, it provides a transparent dumb-terminal method of communication. For MS-DOS-compatible computers, it also provides a tvi-920 emulation.

It also supports the Ward Christiansen file transfer protocol to facilitate the error-free transfer of files between systems both supporting this protocol. The default error-checking method used is checksum. CMODEM will acknowledge CRC error checking and will switch if requested by the other computer. This feature allows the use of CMODEM with most available upload/download facilities.

For those systems that do not support a batch transfer protocol, CMODEM supports an ASCII file collection/dump mode for transfer of text files. In this mode, and in dumb-terminal mode, it supports an X-ON/X-OFF mode of controlling the data transmission.

CMODEM will work at speeds of 300 to 9600 Baud for those systems with asynchronous communications support devices. It is used regularly at 9600 Baud between adjacent machines to transfer data between them.

The CMODEM program with printed manual is available for a retail price of \$50 without source or \$100 with source.

68010 SUPER SLEUTH

A disassembler performs a function opposite to that of an assembler. That is, it allows its user to process an object program, for which no source program is readily available, and to re-create the essentials of the original source program. The resultant source program may then be studied, commented, modified, reassembled, etc. A trivial disassembly may be performed rather simply manually or with the use of a computer by translating the successive bytes of object code into the equivalent mnemonics for a target processor. A really useful disassembly requires the assignment of labels and the ability to determine whether given bytes in the program represent data, addresses, garbage, or instructions. Since a computer can assign labels but cannot classify program contents, disassembly must be interactive to be meaningful.

SUPER SLEUTH is a collection of programs which enables the user to interactively examine Motorola S-format program files on disk. 68010 programs may be disassembled into source code format and the source saved on disk.

Programs in other binary formats may be placed into Motorola S-format with utility programs provided with SUPER SLEUTH. Labels produced by SUPER SLEUTH can be changed globally to labels of your own preference. Cross reference listings of labels can be produced to aid in debugging or modifying the program.

SUPER SLEUTH, with printed manuals, is available in object-only form for a retail price of \$100, or with source for \$200.

PALM BEACH SOFTWARE 7080 HYPOLUXO FARMS RD. LAKE WORTH, FL 33463 (407) 965-2657

CONGRATULATIONS on the purchase of your PT-68K2. I have been selling and using Peripheral Technology products for the past 6 years and my customers and I are very happy with their products, service and support. Since 1978 I have been developing business oriented programs for the Motorola 6800 & 6809 and I am now converting them to the 68000. The following products are available from Palm Beach Software.

EDDI a screen editor and formatter. \$50.00

EDDI has 62 key commands to completely manage the text and screen display, and 22 formatting commands to control the printed output. It is also convenient for use as a program editor supporting line numbers, tab stops and macro keys. A spelling helper, synonym finder and other DOS utilities are available without exiting EDDI. Two banks of memory are supported so that you can edit two files and move data back and forth. DOCUMAKER (a boiler plate document creator) and MERGER (personalized name, text and address insertion) will be available in late 1988.

SPELLB a 160,000 word spelling checker. \$50.00

SPELLB has been around in the 6809 version for 6 years. It is still on version 1.0 and I have only fixed two misspellings.

ASMK a native code assembler. \$25.00

ASMK is a bare bones, native code, 68000 assembler. It does not have macros or conditional assembly and you must have enough memory to hold the complete source and object code. The syntax is similar to other 6800 and 6809 assemblers which were common to the SS-50 world. It is very fast.

SUBCAT a sub-directory manager. \$25.00

SUBCAT is a program that will allow multiple directories; setup in a tree structure on a disk. The disk directory is not altered and is still the primary source of information about each file. Each entry has a 36 byte message area where you can keep a description of the file and its contents. A long block of reverse video is moved to select an entry and now you can execute the following operations on the selected file. Assemble, copy, delete, edit, get or execute, help, kill, install, load, enter message, rename and view. Other operations; backup, execute DOS command, scan, update, zap and exit can be done from any position. Two other commands are user definable. In essence you can create a tree like catalog structure and then manipulate the files in these catalogs as an independent directory.

KRACKER a disassembler program. \$25.00

KRACKER will allow you to scan through a program to create a memory map of code, text, tables, etc. When the memory map is completed you can create a source code file to disk, printer, or terminal.

LOGIC a schematic drawing prog. graphic screen only \$25.00

MOZART a music editor (requires special card) \$25.00

The following program is underdevelopment and is expected to be released in the third quarter of 1988.

PARASOL a complete integrated business system.

PARASOL is a alternate tasking, memory management, menu driven, umbrella or shell program for the PT-68K2. The main menu can be reentered from any program, where you may select another task. This task may also reenter the main menu and select again. When you exit a task you will be returned to the previous task at the exact point where you left with all pointers, flags and screens restored. At the present time the following programs and desk accessories are either working or under development. Multituser version can be expected in the fall of 1988.

APPOINTMENT Calendar

BILLS PAYABLE

COMMUNICATIONS PACKAGE (Email, Modem Management, etc)

DIARY (Note pad data base organized by name & time)

EQUIPMENT (Data base of property (depreciation, loans & borrowings))

FILE ROOM (Main data base organized as Cabinets (folders), Drawers (directories), Folders (sub-directories), Documents (files). After a document has been selected you will then have the option to edit, view, rename, copy, send, delete, assemble, etc.

GENERAL LEDGER (complete double entry accounting package)

HOUSEKEEPING (daily journal & posting to all accounts)

INVOICE GENERATOR (create invoices and delivery tickets)

JOB JAR (data base of things to do)

COST ACCOUNTING (integrated with bills received & payroll)

TIME LOG & MANAGEMENT (professional time monitor for billing)

FILE MAINTENANCE (where all boo-boos are corrected)

NAME & ADDRESSES (main data base as most other files are referenced to the personal file)

PURCHASE ORDERS (data base of orders placed)

PAYROLL (1000 employees, checks, monthly, quarterly, yearly reports and W2's)

EMERGENCY EXIT (means of shutdown procedures)

RECEIVED ON ACCOUNT (integrated with invoice generator so keep record of charges and payments, statements, past due report)

SPREAD SHEET

TELEPHONE & DIALER (dial telephone and keep record of time & charges)

UTILITY ROUTINES

INVENTORY (manage the inventory integrated with invoice generator)

EXPENSE ACCOUNTING (log & manage expense accounts)

Palm Beach Software programs are written in assembly language for speed and efficient use of ram and disk memory. They are not copy protected and the source code is made available for the modules that are user defined. (Printer drivers, Terminal drivers, Modem drivers and Key selection tables) A smart terminal is required for our business software. I recommend the Televideo 905, but the Wyse WY-50 is also a good choice. A smart terminal emulator program has been written for the monographics card and AT style keyboard. However, if you plan in the future to use the business programs, either single or multiuser, a smart terminal should be used. Please include a description of your system with each order. Software will be distributed on 5 1/4" disks, Single Sided, Double Density, and 96 TPI unless a special request is made.

Happy Hacking,

Dan Farnsworth

GESPAC Inc.
50 West Hoover Ave.
Mesa, Arizona 85210
Tel. (602) 962-5559
Fax. (602) 962-5750

Reader Contact: Don Bizios
Editorial Contact: Cosma Pabouctsidis

DO NOT RELEASE UNTIL 12/15/88

80286 G-64 BOARDS SET HELPS BUILD RUGGED

IBM PC AT COMPATIBLE SYSTEM

Mesa, AZ, November 1, 1988--GESPAC has released a board set implementing the Intel 80286 16-bit processor, VGA graphics, and disk controller that is compatible with the IBM AT running at 10 MHz. The three boards are each built under the single height Eurocard form factor, use the reliable DIN 41612-C connector, and are designed to operate on the G-64 bus. The unique mechanical characteristics and bus extension capability of the board set makes it ideal for use in applications where a harsh or vibrating environment would prohibit the use of a regular office PC or compatible.

The first of the boards is a highly integrated industrial implementation of Intel's high performance 80286 16-bit microprocessor. The microprocessor has 512K of fast access on-board dynamic memory and sockets for the EPROM BIOS. For math intensive operations, an 80287 arithmetic coprocessor can be added. In addition, the board includes two RS-232 serial ports, three timers, a battery backed-up clock and calendar, as well as a powerful interrupt controller.

The second member of the system is an Enhanced Graphics Adapter (EGA) that features an integral AT keyboard interface. This brings PC compatible color graphics to the industrial environment. The third member of the system is a hard disk controller to be used under the ST-506 interface under the G-64 bus, giving the user the option of using a vast number of hard disks, as well as capacities at a low cost.

All three boards communicate through a standard G-64 bus backplane. The G-64 bus is a 16/32-bit open bus architecture, featuring the single height Eurocard board format (4" by 6.25"), that is widely used in industrial and embedded systems application. The bus interface allows the board set to be expanded with any of over 150 I/O, data acquisition or controller modules available for the bus.

The board set has been demonstrated to run all the most popular programs written for the IBM PC AT.

Commodore NEWS

For further information
contact Valerie Bellofatto or
Lori Cross at Fleishman-Millard
(213) 629-4974

COMMODORE INTRODUCES AMIGA[®] WITH 68020

Amiga 2500 features doubled processor speed, math-co-processor and 2 MB of 32-bit RAM for workstation-level applications

LAS VEGAS, November 14, 1988 -- Commodore Business Machines, Inc. today introduced the Amiga 2500 designed for the needs of graphics, animation and video professionals. Based on the original Amiga 2000 introduced a year ago, the Amiga 2500 is configured with an A2620/2 co-processor card that comes standard with the Motorola[®] 68020 processor, 2 MB of 32-bit RAM (expandable to 4 MB) and a 68881 math-co-processor. These high-performance features enable the new Amiga system to perform at the workstation-level speeds required by many of the new second generation graphics-based Amiga applications.

"The Amiga 2500 is a natural progression of the Amiga technology," said Joel Shusterman, Commodore vice president of marketing. "The Amiga 2500 has no equal in its price/performance class and offers all of the original design benefits of the Amiga 2000 as an expandable, multi-tasking, multi-processing, multi-operating system machine."

The advanced graphics capability of the Amiga series is ideally suited to color desktop publishing and presentation, 3-D solid modeling and professional video applications. With the additional speed of the Amiga 2500, color rendering time of 3-D graphics and re-calculation time for modifying full-color desktop publishing pages can be reduced substantially. Performance increases can be up to 400 percent.

Standard on the Commodore Amiga 2500 is a Motorola 68020-based co-processor card (A2620/2) running at 14.26 Mhz with 2 MB of 32-bit RAM, (expandable to 4 MB of 32-bit RAM); a built-in 3.5-inch floppy disk drive; a pre-configured, high-performance 40 MB hard disk drive and hard disk controller; custom sound, animation and graphics chips; RS232 serial and parallel connectors; and two RCA-type audio output jacks. Also available is an MS-DOS[®] compatible Bridgeboard allowing the Commodore Amiga to run MS-DOS compatible software under Amiga control.

The open architecture of the Commodore Amiga allows extensive internal expansion with multi-processor, multi-DOS options. Contained in the Amiga 2500 are seven full-size internal expansion slots which include a combination of Amiga, standard PC XT/AT and dual purpose slots; a CPU expansion slot and a video expansion slot.



Product News

Prepared By:
Shohat & Davis PR
2959 S. Winchester Blvd, Campbell, CA 95008
Murry Shohat (408) 379-7434

Contact:
FORCE USA: Wayne Fischer (408) 370-6300
FORCE GmbH: Anton Nausch (089) 600-910

Extraordinary 68030-based Single-Board Computer Offers Large Number of Functions at Low Price

Cost/Performance Breakthrough with Options on Speed, Memory, SCSI, Floppy and Ethernet

CAMPBELL, CA., October 18, 1988 — A new high-end 68030-based single board computer introduced today brings cost-efficient modularity to high performance 32-bit applications. The CPU-37 offers more functionality than most single-board VMEbus-based computers. The board offers a range of options on CPU speed, on-board memory and floppy, SCSI and Ethernet control. When fully configured, this single-board computer replaces solutions that use up to four boards. FORCE has priced the CPU-37 below any comparable single or multi-board offering.

The CPU-37 also offers an ASIC solution to VMEbus interface and control. "Using our advanced VME/PLUS™ architecture, we married high-density application-specific gate array technology with precision surface-mount board manufacturing," said Wayne Fischer, Director of Marketing. "We're able to offer more functions in a single board computer for less dollars, thanks to VME/PLUS. The CPU-37 will serve the low-to-middle reaches of the demanding high-performance market segment. Customers can select CPU speeds of either 16.7 or 25 MHz, on-board DRAM of either 1 or 4 Mbytes, and say yes or no to Ethernet."

Standard Features Will Attract Mid-Range, Embrace High-End Applications

The most basic CPU-37 is priced as low as \$3,990 but offers features associated with boards costing hundreds more. A fully configured CPU-37 is priced at \$5,890 and includes all features described below. "We're bound to attract applications that up until now have been too cost-sensitive to move to a 68030-based CPU," said Fischer. "And, high-end buyers will see the CPU-37 as a price-performance breakthrough."

The CPU-37's ranges of features include:

- 68030 microprocessor, 16.7 or 25 MHz operation; 2 E²PROM sockets
- 68882 floating point coprocessor, 16.7 or 25 MHz operation
- 1 or 4 Mbytes of DRAM, 0 wait states at 16.7 MHz, 1 wait state at 25 MHz
- Real time clock with on-board battery backup

- 3 serial ports for RS232/RS422 operation available via 9-pin front panel connectors. First 2 ports can provide synchronous operation based on the 68562 dual universal serial communications controller chip; 3rd port employs the 68901 multi-function peripheral chip.
- 1 Parallel Interface/Timer provides one 24-bit and four 8-bit timers
- SCSI interface (via MB87031 host adapter chip)
- Floppy Disk Controller (SA460) interface (via WD1772 controller chip)
- Optional Ethernet transceiver interface (front panel access) with 64 Kbyte dedicated buffer, based on the AMD 7990 LANACE chip
- VME/PLUS architecture embodied in PGA-001, a 135-pin CMOS gate array with 1.5 micron feature size. The array consumes less than 120 milliwatts, features remarkably low gate delays (1.4 ns) and is capable of an internal toggle frequency of 200 MHz and external toggle frequency of 70 MHz. This array provides VMEbus interface and control functions including DSACK generation, bus error generation, system reset, bus clock and all on-board control logic.
- A32/24/16, D32/24/16/8 VMEbus master/slave interface
- Slot 1 Control functions (SYSLOCK, arbiter, etc.)

Software Includes Free OS Kernel, Wide 3rd-Party Compatibility

Enhancing the usefulness of the CPU-37 is VMEPROM, a free real-time operating system kernel that also includes a monitor and debugger. It is installed on the board in EPROM, yielding operational capability as soon as the CPU-37 is installed on an active backplane.

VMEPROM is based on PDOS, a popular operating system from Eyring Research Institute.

The CPU-37 is also compatible with third-party real-time systems and kernels, including UNIX compatible products. Current third-party ports include OS-9 (Microware), VRTX32 (Ready Systems), pSOS (Software Components Group), VxWorks (Wind River Systems) and UNIXEX (TSC).

Support for Ethernet's TCP/IP protocol is planned for 1Q89.

Price & Availability

The CPU-37 has entered the shipping pipeline and is available for immediate delivery. Equipped with a 16.7MHz 68030 and 68882 math coprocessor, 1MB of memory, floppy and SCSI controllers, the CPU-37X is priced at \$3,990 (1-9). A 25MHz version with 4MB of memory, floppy, SCSI and Ethernet controllers is priced at \$5,890. The product is shipped in a new FORCE package that emphasizes static protection.

About Force Computers

The leading independent designer and manufacturer of VMEbus products, Force is now in its sixth year. The company has completed 23 consecutive quarters of profitable operation. Force is headquartered in Campbell, California with subsidiaries in West Germany, France and the United Kingdom. Sales, service and product support are provided on a worldwide basis.

Classifieds As Submitted - No Guarantees

MUSTANG-020 16Mhz with 68881. OS9 Professional Package & C \$2500.

S+System with Cabinet, 20 Meg Hard Disk & 8" Disk Drive with DMAF3 Controller Board. 1-X12 Terminal \$2900.

HARD DISK 10 Megabyte Drive - Seagate Model #412 \$275.

3-Dual 8" drive enclosure with power supply. New in box. \$125 each.

5-Siemens 8" Disk Drive, \$100 each.

Tano Outpost II, 56K, 2 5" DSDD Drives, FLEX, MUMPS, \$250.

QUME QVT-102 terminal, like new, amber screen \$250, or best offer.

SWTPC S/09 with Motorola 128K RAM, 1-MPS2, 1-Parallel Port, MP-09CPU Card- \$490 complete.

Tom (615) 842-4600 M-F 9AM to 5PM EST

Used Systems, Components & Software: 6809 GIMIX, ELEKTRA, 6800 GIMIX, SWTP & SSB, 5-1/4 DSDD Drives, Terminals, 300-9600 Cassette I/F & Backup Software, TSC, CSC, ETC. FLEX 2/9 Software FOR GIMIX, ELEKTRA & SWTP. OS9LI FOR ELEKTRA & HELIX. Credit Cards Accepted. Send SASE For List To George Ingram 26W482 NATIONAL ST., WHEATON, IL., 60188 or CALL 312-653-0360 CST M-TH 7:30PM-10PM, S-S 7:30AM-10PM.

NEW!

OmegaSoft Pascal for the 68020/68881

P20K is a Pascal package that will generate code for all of the 68000 series processors, including the 68881 coprocessor. P20K will run on any 68000 series computer running the OS-9/68000 (Microware) or PDOS (Eyring Research) operating systems with 512K or more free memory.

The base package (P20K-B) includes the Compiler, Relocatable Macro Assembler, Linking Loader, Screen Editor, Pascal Shell, Linkage Creator, Host Debugger, Configuration manager, Installation program, and Patch utility. A new feature in this compiler is the ability to either link in the parts of the runtime needed by the program, or to use trap handlers for runtime access, to share the runtime library between programs. Complete operating system interface is also included using pascal procedures and functions. The host debugger allows debugging at both the Pascal and assembly language levels of programs that run on the host operating system. Price for the base package is \$575.

The runtime source code option (P20K-R) is available for \$100 and includes source code for the operating system interface routines as well as pascal runtime.

The Utility source option (P20K-S) is available for \$275 and includes source code for the Screen Editor, Pascal Shell, Host Debugger, Patch utility, and Configuration manager.

The Target debugger option (P20K-T) is \$225 and includes object and source code. This program allows Pascal level and assembly level debugging in a system without operating system, by using a serial link connected to the host computer.

Prices do not include shipping charges. Master-Card and Visa accepted. OmegaSoft is a registered trademark of Certified Software Corporation.

Gespac SA, 3, Chemin des Aulx, CH-1228, Geneva/Plan-les-Quates, Switzerland. TEL 022-713400, TLX 429889

Elsoft AG, Zeilweg 12, CH-5405 Baden-Dättwil, Switzerland, TEL 056-833377, TLX 828275

RCS Microsystems Ltd., 141 Uxbridge Road, Hampton Hill, Middlesex, England. TEL 01-9792204, TLX 8951470

Byte Studio Borken, Butenwall 14, D-4280 Borken, West Germany. TEL 02861-2147, TLX 813343

Eltec Elektronik GmbH, Galileo-Galilei-Straße, 6500 Mainz 42, Postfach 65, West Germany TEL 06131-50031, TLX 4187273

PEP Elektronik Systeme GmbH, Am Klosterwald 4 D-8950 Kaufbeuren, West Germany TEL 06341-8974, TLX 541233

**CERTIFIED
SOFTWARE
CORPORATION**

P.O. BOX 70, RANDOLPH, VT 05060 USA
TELEPHONE: (802) 728-4062
FAX: (802) 728-4126

FLEX™/SK-DOS™/MS-DOS™

Transfer Utilities

For 68XXX and CoCo* OS-9™ Systems

Now READ - WRITE - DIR - DUMP - EXPLORE

FLEX, SK-DOS & MS-DOS Disk

These Utilities come with a rich set of options allowing the transfer of text type files from/to FLEX & MS-DOS disks.

*CoCo systems require the D.P. Johnson SDISK utilities and OS-9 and two drives of which one must be a "host" floppy.

CoCo Version: \$69.95

68XXX Version \$99.95

S.E. Media

615 842-6809

PO Box 849, Hixson, TN 37343

MC/Visa

SK*DOS®/68K

Read the fine print to see what's in SK*DOS/68K:

☐ Full DOS documentation plus on-line help ☐ Multiple directories
☐ User-installable device drivers ☐ Install up to 8 different I/O devices ☐ Keyboard type-ahead ☐ Print-screen ☐ Virtual (RAM) disk ☐ Disk cache ☐ Up to 10 drives ☐ 5¼" or 3½" floppy drives ☐ Hard drives to 64 megabytes each ☐ I/O redirection to drives or I/O ☐ Time/date stamping of files ☐ File or disk write protect (even hard disk) ☐ Batch files ☐ Support for 68000, 68010, 68020 ☐ Monochrome or color video board support ☐ Read and write MS-DOS disk files ☐ 6809 Emulator ☐ Powerful utilities such as copy-by-date, undelete, show differences between files, prompted delete, text file browse, and more - all included ☐ Simple Basic included ☐ Fast assembler included ☐ Line editor included ☐ User support via newsletter and BBS ☐ Available software: C compiler, full Basic, screen editors, disassemblers, cross-assemblers, spelling checker, text formatter, music editor, hard disk manager, ROM-based debugger, modem communications programs, etc. More compilers coming. ☐ (Some features may not be implemented in all hardware manufacturers' implementations.)

Individual copies of SK*DOS/68K are \$140; less in quantity or when bundled with hardware. Send for our 6809 / 68K hardware and software catalog. Also available as part of our hardware/software educational course.



Software Systems Corp.
 P. O. Box 209J
 Mt. Kisco, NY 10549
 (914) 241-0287
 BBS (914) 241-3307 • Fax (914) 241-8607



APPLE MACINTOSH™ USERS



**Save over a \$1,000.00
 on PostScript
 Laser Printers!
 Faster - Finer Quality
 than the original Apple
 LaserWriter!
 New & Demos
 Cartridges-new-rebuilds
 -colors-**

**In Chattanooga Call:
 615 842-1600
 QMS-Authorized**

Data-Comp Division
 A Decade of Quality Service
 Systems World Wide
 Computer Publishing, Inc. 5600 Cassandra Smith Road
 Telephone 615-842-4801 • Telex 511 600-6630 • Hixson, TN 37343

SOFTWARE

68000 C CROSS-COMPILER

\$100 - SKDOS, MSDOS, UNIX, XENIX (OBJECT ONLY)

Accepts K&R C language, generates 68000 assembly code
 includes 68010 cross-assembler, libraries provided for SKDOS, but may be modified.

CROSS-ASSEMBLERS WITH MACRO CAPABILITIES

EACH \$50-FLEX, OS9, UNIX, FLEX, MSDOS, UNIX, SKDOS, XENIX \$1500 ALL \$1200

Specify: 1600s, 6502, 6801/11, 6804, 6805, 6809, Z8, Z80, 8048, 8051, 8085, 68010, 32000
 Modular cross-assemblers in C, with full/linked utilities. Sources for additional \$50 each, \$100 for 3, \$300 for all.

C/MODEM TELECOMMUNICATIONS PROGRAM

\$100-MSDOS, SKDOS, UNIX, FLEX, OS9, XENIX, UNIX, OBJECT ONLY: EACH \$50

Menu-driven with terminal mode, file transfer, M0DEM7, XON/XOFF, etc.

SUPER SLEUTH DISASSEMBLERS

EACH \$99-FLEX \$101-OS9 \$100-UNIX/FLEX OBJECT ONLY: EACH \$50 FLEX, OS9, COCO

Interactively generate source on disk with labels, includes xref, binary editing
 Specify 6800, 1.2, 3.5, 8, 9/6502 version or Z80/8080, 5 version
 COCO DOS available in 6800, 1.2, 3.5, 9/6502 version (not Z80/8080, 5) only
 68010 version \$100-FLEX, OS9, UNIX, FLEX, MSDOS, UNIX, SKDOS, XENIX

DEBUGGING SIMULATORS FOR POPULAR 8-BIT CPUs

EACH \$75-FLEX \$100-OS9 \$80-UNIX/FLEX OBJECT ONLY: EACH \$50 COCO FLEX, COCO OS9

Interactively simulate processors, includes disassembly formatter, binary editing
 Specify for 6800/1, (14)6805, 6502, 6809 OS9 only, Z80 FLEX only

ASSEMBLER CODE TRANSLATORS FOR 6502, 6800/1, 6809

6502 to 6809 \$75-FLEX \$86-OS9 \$80-UNIX/FLEX
 6800/1 to 6809 & 6809 to pos.ind. \$50-FLEX \$75-OS9 (only \$60-UNIX/FLEX)

FULL-SCREEN XBASIC PROGRAMS with cursor control AVAILABLE FOR FLEX, UNIX, AND MSDOS

Display Generator / Overmaster	\$50 w/source, \$25 without
Mailing List System	\$100 w/source, \$50 without
Inventory with MRP	\$100 w/source, \$50 without
Tabula Rasa Spreadsheet	\$100 w/source, \$50 without

DISK AND XBASIC UTILITY PROGRAM LIBRARY \$50-FLEX \$10-UNIX/FLEX/MSDOS

Edit disk sectors, sort directories, maintain master catalog, do disk sorts, resequence some or all BASIC programs, xref BASIC programs, etc. non-FLEX versions include sort and resequence only

PROFESSIONAL SERVICES FOR THE COMPUTING COMMUNITY

CUSTOMIZED PROGRAMMING

We will customize any of the programs described in this advertisement or in our brochure for specialized customer use or to cover new processors; the charge for such customization depends upon the marketability of the modifications.

CONTRACT PROGRAMMING

We will create new programs or modify existing programs on a contract basis, a service we have provided for over twenty years; the companies on which we have performed contract programming include most popular models of mainframes, including IBM, Burroughs, Univac, Honeywell, most popular models of minicomputers, including DEC, IBM, DG, HP, AT&T, and most popular brands of microcomputers, including 6800V1, 6809, Z80, 6502, 68010, using most appropriate languages and operating systems, on systems ranging in size from large telecommunications to single board controllers; the charge for contract programming is usually by the hour or by the task.

CONSULTING

We offer a wide range of business and technical consulting services, including seminars, advice, training, and design, on any topic related to computers; the charge for consulting is normally based upon time, travel, and expenses.

Computer Systems Consultants Inc.

1454 Little Lane
 Conyers, Georgia 30207
 (404) 483-4370 • (404) 483-1717

Contact us about catalog, dealer, discounts, and services. Most programs in source give computer, OS, disk size. 25% off multiple purchases of same program on one order. VISA and MASTER CARD accepted. Add GA sales tax (if in GA) and 5% shipping. (UNIX/FLEX on Technical Systems Consultants; OS9 Microwave COCO Tandy; MSDOS Microsoft; SKDOS Stack Software)

Clearbrook Software Group

(604)853-9118



CSG IMS is *THE* full featured relational database manager for OS9/OSK. The comprehensive structured application language and B+ Tree index structures make **CSG IMS** the ideal tool for file-intensive applications.

CSG IMS for CoCo2/3 OS9 L1/2 (single user)	\$169.95
CSG IMS for OS9 L2 or 68000 (multi user)	\$495.00
CSG IMS demo with manual	\$30

MSF - MSDos File Manager for CoCo 3/OS9 Level 2
allows you to use MSDos disks directly under OS9.
Requires CoCo 3, OS9 L2, SDISK3 driver \$45.00

SERINA - System Mode Debugger for OS9 L2

allows you to trace execution of any system module, set break points, assemble and disassemble code and examine and change memory.

Requires CoCo3 or Gimix II, OS9 L2 & 80 col. terminal \$139.00

ERINA - Symbolic User Mode Debugger for OS9

lets you find bugs by displaying the machine state and instructions being executed. Set break points, change memory, assemble and disassemble code.

Requires 80 column display, OS9 L1/2 \$69.00

Shipping: N. America - \$5, Overseas - \$10
Clearbrook Software Group P.O. Box 8000-499, Sumas, WA 98295
OS9 is a trademark of Microware Systems Corp., MSDos is a trademark of Microsoft Corp.

SPECIAL

ATARI™

&

OS-9™

NOW!

If you have either the
Atari 520 or 1040 -
you can take
advantage of the
"bargain of a lifetime"
OS-9 68K and BASIC
all for the low, low price of:

\$150.00

Call or Write

S.E. Media

5900 Cassandra Smith Rd.

Hixson, TN 37343

615 842-4601

ATARI & AMIGA CALL

As most of you know, we are very sensitive to your wishes, as concerns the contents of these pages. One of the things that many of you have repeatedly written or called about is coverage for the **Atari & Amiga™** series of 68000 computers.

Actually we haven't been too keen on those systems due to a lack of serious software. They were mainly expensive "game-toy" systems. However, recently we are seeing more and more honest-to-goodness serious software for the Atari & Amiga machines. That makes a difference. I feel that we are ready to start some serious looking into a section for the Atari & Amiga computers. Especially so since OS-9 is now running on the Atari (review copy on the way for evaluation and report to you) and rumored for the Amiga. Many of you are doing all kinds of interesting things on these systems. By sharing we all benefit.

This I must stress - Input from you on the Atari & Amiga. As most of you are aware, we are a "contributor supported" magazine. That means that **YOU** have to do your part. Which is the way it has been for over 10 years. We need articles, technical, reviews of hardware and software, programming (all languages) and the many other facets of support that we have pursued for these many years. Also I will need several to volunteer to do regular columns on the Atari & Amiga systems. Without constant input we can't make it fly! So, if you do your part, we certainly will do ours. How about it, drop me a line or give me a phone call and I will get additional information right back to you. We need your input and support if this is to succeed!

DMW

THE 6800-6809 BOOKS

..HEAR YE.....HEAR

OS-9™ User Notes

By: Peter Dibble

The publishers of 68' Micro Journal are proud to make available the publication of Peter Dibble's **OS9 USER NOTES**

Information for the BEGINNER to the PRO,
Regular or CoCo OS9

Using OS9

HELP, HINTS, PROBLEMS, REVIEWS, SUGGESTIONS, COMPLAINTS,
OS9 STANDARDS, Generating a New Bootstrap, Building a
new System Disk, OS9 Users Group, etc.

Program interfacing to OS9

DEVICE DESCRIPTORS, DIRECTORIES, "FORKS", PROTECTION,
"SUSPEND STATE", "PIPES", "INPUT/OUTPUT SYSTEM", etc.

Programming Languages

Assembly Language Programs and Interfacing; Basic09, C,
Pascal, and Cobol reviews, programs, and uses; etc.

Disks Include

No typing all the Source Listings in. Source Code and,
where applicable, assembled or compiled Operating
Programs. The Source and the Discussions in the
Columns can be used "as is", or as a "Starting Point"
for developing your OWN more powerful Programs.
Programs sometimes use multiple Languages such as a
short Assembly Language Routine for reading a
Directory, which is then "piped" to a Basic09 Routine
for output formatting, etc.

BOOK \$9.95

Typeset -- w/ Source Listings
(3-Hole Punched; 8 x 11)

Deluxe Binder - - - - - \$5.50

All Source Listings on Disk

1-8" SS, SD Disk - - - - \$14.95

2-5" SS, DD Disk - - - - \$24.95

Shipping & Handling \$3.50 per Book, \$2.50 per Disk set

Foreign Orders Add \$4.50 Surface Mail
or \$7.00 Air Mail

If paying by check - Please allow 4-6 weeks delivery

* All Currency in U.S. Dollars

Continually Updated In 68 Micro Journal Monthly

**Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343**



*FLEX is a trademark of Technical Systems Consultants

*OS9 is a trademark of Microware and Motorola

*68' Micro Journal is a trademark of Computer Publishing Inc.

FLEX™ USER NOTES

By: Ronald Anderson

The publishers of 68 MICRO JOURNAL are proud to make available the publication of Ron Anderson's **FLEX USER NOTES**, in book form. This popular monthly column has been a regular feature in 68' MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. The most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINTC4	Parallel printer driver, without PFLAG — ASM
SETC5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASIC

NOTE: .C1, .C2, etc. = Chapter 1, Chapter 2, etc.

**Over 30 TEXT files included is ASM (assembler)-PASCAL-
PIC (position independent code) TSC BASIC-C, etc.

Book only: \$7.95 + \$2.50 S/H

With disk: 5" \$20.90 + \$2.50 S/H

With disk: 8" \$22.90 + \$2.50 S/H



(615) 842-4601

Telex 5106006630

!!! Subscribe Now !!! 68 MICRO JOURNAL

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Mastercard ☐ VISA ☐

Card # _____ Exp. Date _____

For 1 Year 2 Years 3 Years _____

Enclosed: \$ _____

Name _____

Street _____

City _____ State _____ Zip _____

Country _____

My Computer Is: _____

Subscription Rates

U.S.A.: 1 Year \$24.50, 2 Years \$42.50, 3 Years \$64.50

*Foreign Surface: Add \$12.00 per Year to USA Price.

*Foreign Airmail: Add \$48.00 per Year to USA Price.

*Canada & Mexico: Add \$9.50 per Year to USA Price.

*U.S. Currency Cash or Check Drawn on a USA Bank !

68 Micro Journal
5900 Cassandra Smith Rd.



POB 849
Hixson, TN 37343



Telephone 615 842-4600
Telex 510 600-6630

Reader Service Disks

- Disk- 1 Filesort, Minicat, Minicopy, Minifms, **Lifetime, **Poetry, **Foodlist, **Diet.
- Disk- 2 Diskedit w/ inst. & fixes, Prime, *Pnnod, **Snoopy, **Football, **Hexpaw, **Lifetime.
- Disk- 3 Cbug09, Sec1, Sec2, Find, Table2, Intext, Disk-exp, *Disksave.
- Disk- 4 Mailing Program, *Findai, *Change, *Testdisk.
- Disk- 5 *DISKFIX 1, *DISKFIX 2, **LETTER, **LOVESIGN, **BLACKJAK, **BOWLING.
- Disk- 6 **Purchase Order, Index (Disk file indx).
- Disk- 7 Linking Loader, Rload, Iarkness.
- Disk- 8 Ctest, Lanpher (May 82).
- Disk- 9 Datecopy, Diskfix9 (Aug 82).
- Disk-10 Home Accounting (July 82).
- Disk-11 Dissembler (June 84).
- Disk-12 Modem68 (May 84).
- Disk-13 *Iniunf68, Testmf68, *Cleanup, *Diskalign, Help, Date.Txt.
- Disk-14 *Init, *Test, *Tenninal, *Find, *Diskedit, Init.Lib.
- Disk-15 Modem9 + Updates (Dec. 84 Gilchrist) to Modem9 (April 84 Commo).
- Disk-16 Copy.Txt, Copy.Doc, Cat.Txt, Cat.Doc.
- Disk-17 Match Utility, RATBAS, A Basic Preprocessor.
- Disk-18 Parse.Mod, Size.Cmd (Sept. 85 Armstrong), CMDCODE, CMD.Txt (Sept. 85 Spray).
- Disk-19 Clock, Date, Copy, Cat, PDEL, Asm & Doc., Errors.Sys, Do, Log.Asm & Doc.
- Disk-20 UNIX Like Tools (July & Sept. 85 Taylor & Gilchrist). Dragon.C, Grep.C, L.S.C, FDUMP.C.
- Disk-21 Utilities & Games - Date, Life, Madness, Touch, Goblin, Starshot, & 15 more.
- Disk-22 Read CPM & Non-FLEX Disks. Fraser May 1984.
- Disk-23 ISAM, Indexed Sequential file Accessing Methods, Condon Nov. 1985. Extensible Table Driven. Language Recognition Utility, Anderson March 1986.
- Disk-24 68' Micro Journal Index of Articles & Bit Bucket Items from 1979 - 1985, John Current.
- Disk-25 KERMIT for FLEX derived from the UNIX ver. Burg Feb. 1986. (2)-5" Disks or (1)-8" Disk.
- Disk-26 Compacta UniBoard review, code & diagram, Burlison March '86.
- Disk-27 ROTABIT.TXT, SUMSTEST.TXT, CONDATA.TXT, BADMEN.TXT.
- Disk-28 CT-82 Emulator, bit mapped.
- Disk-29 **Star Trek
- Disk-30 Simple Winchester, Dec. '86 Green.
- Disk-31 *** Read/Write MS/PC-DOS (SK-DOS)
- Disk-32 Heir-UNIX Type upgrade - 68MJ 2/87
- Disk-33 Build the GT-4 Terminal - 68MJ 11/87 Condon.
- Disk-34 FLEX 6809 Diagnostics, Disk Drive Test, ROM Test, RAM Test - 68MJ 4/88 Koipi.

NOTE:

This is a reader service ONLY! No Warranty is offered or implied, they are as received by 68' Micro Journal, and are for reader convenience ONLY (some MAY include fixes or patches). Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other. Software is available to cross-assemble all.

* Denotes 6800 ** Denotes BASIC

*** Denotes 68000 - 6809 no indicator.



8" disk \$19.50
5" disk \$16.95



Shipping & Handling -U.S.A. Add: - \$3.50
Overseas add: \$4.50 Surface - \$7.00 Airmail

68 MICRO JOURNAL

5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4600 - Telex 510 600-6630

K-BASIC™

The Only 6809 BASIC to Binary Compiler for OS-9
FLEX or SK*DOS

Even runs on the 68XXX SK*DOS Systems*

*Hundreds Sold at
Suggested Retail:*

~~\$199.00~~

• 6809 - OS-9™ users can now transfer their FLEX™ Extended BASIC (XBASIC) source files to OS-9, compile with the OS-9 version and run them as any other OS-9 binary "CMD" program. Much faster than BASIC programs.

• 6809 - FLEX users can compile their BASIC source files to a regular FLEX ".CMD" file. Much faster execution.

• 68XXX - SK*DOS™ users running on 68XXX systems (such as the Mustang-08/A) can continue to execute their 6809 FLEX BASIC and compiled programs while getting things ported over to the 68XXX. SK*DOS allows 6809 programs to run in emulation mode. This is the only system we know of that will run both 6809 & 68XXX binary files.

K-BASIC is a true compiler. Compiling BASIC 6809 programs to binary command type programs. The savings in RAM needed and the increased speed of binary execution makes this a must for the serious user. And the price is now RIGHT!

Don't get caught up in the "Learn a New Language" syndrome - Write Your Program in BASIC, Debug It in BASIC and Then Compile it to a .CMD Binary File.

For a LIMITED time
save over 65%...
This sale will not be
repeated after it's
over! *

SALE SPECIAL:

\$69.95

SPECIAL Thank-You-Sale

Only From:

CPI

S.E. Media™

5900 Cassandra Smith Rd.

Hixson, Tn 37343

Telephone 615 842-6809

Telex 510 600-6630

A Division of Computer Publishing Inc.
Over 1,200 Titles - 6800-6809-68000

* K-BASIC will run under 68XXX SK*DOS in emulation mode for the 6809.

Price subject to change without notice.

PT-68000 SINGLE BOARD COMPUTER

The PT68K2 is Available in a Variety of Formats
From Basic Kits to Completely Assembled Systems

BASIC KIT (8 MHZ) - Board, 68000,
HUMBUG MONITOR + BASIC in ROM,
4K STATIC RAM, 2 SERIAL PORTS, all
Components **\$200**

PACKAGE DEAL - Complete Kit with
Board 68000 10 MHZ, SK-DOS, 512K
RAM, and all Necessary Parts **\$575**

ASSEMBLED BOARD (12 MHZ)
Completely Tested, 1024K RAM,
FLOPPY CONTROLLER, PIA, SK-DOS
\$899

ASSEMBLED SYSTEM - 10 MHZ
BOARD, CABINET POWER SUPPLY,
MONITOR + KEYBOARD, 80 TRACK
FLOPPY DRIVE, CABLES **\$1299**
For A 20 MEG DRIVE, CONTROLLER
and CABLES **Add \$295**

PROFESSIONAL OS9 \$500



FEATURES

- MC68000 Processor, 8 MHZ Clock (optional 10, 12.5 MHZ)
- 512K or 1024K of DRAM (no wait states)
- 4K of SPAM (6116)
- 32K, 64K or 128K of EPROM
- Four RS-232 Serial Ports
- Floppy disk controller will control up to four 5 1/4", 40 or 80 track.
- Clock with on-board battery.
- 2 - 8 bit Parallel Ports
- Board can be mounted in an IBM type PC/XT cabinet and has a power connector to match the IBM type power supply.
- Expansion ports - 6 IBM PC/XT compatible I/O ports. The HUMBUG™ monitor supports monochrome and/or color adaptor cards and Western Digital winchester interface cards.

PERIPHERAL TECHNOLOGY

1480 Terrell Mill Rd., Suite 870
Marietta, Georgia 30067
404/984-0742

VISA/MASTERCARD/CHECK/C.O.D.

Send For Catalogue

For Complete Information On All Products

**SK-DOS is a Trademark of
STAR-K SOFTWARE SYSTEMS CORP.
**OS9 is a Trademark of Microware

DATA-COMP

SPECIAL

Heavy Duty Power Supplies



For A limited time our HEAVY DUTY SWITCHING POWER SUPPLY. These are BRAND NEW units. Note that these prices are less than 1/4 the normal price for these high quality units.

Make: Boschert

Size: 10.5 x 5 x 2.5 inches

Including heavy mounting bracket and heatsink.

Rating: in 110/220 volts ac (strap change) Out: 130 watts

Output: +5v - 10 amps
+12v - 4.0 amps
+12v - 2.0 amps
-12v - 0.5 amps

Mating Connector: Terminal strip

Load Reaction: Automatic short circuit recovery

SPECIAL: \$59.95 each

2 or more \$49.95 each

Add: \$7.50 each S/H

Make: Boschert

Size: 10.75 x 6.2 x 2.25 inches

Rating: 110/220 ac (strap change) Out: 81 watts

Outputs: +5v - 8.0 amps
+12v - 2.4 amps
+12v - 2.4 amps
+12v - 2.1 amps
-12v - 0.4 amps

Mating Connector: Molex

Load Reaction: Automatic short circuit recovery

SPECIAL: \$49.95 each

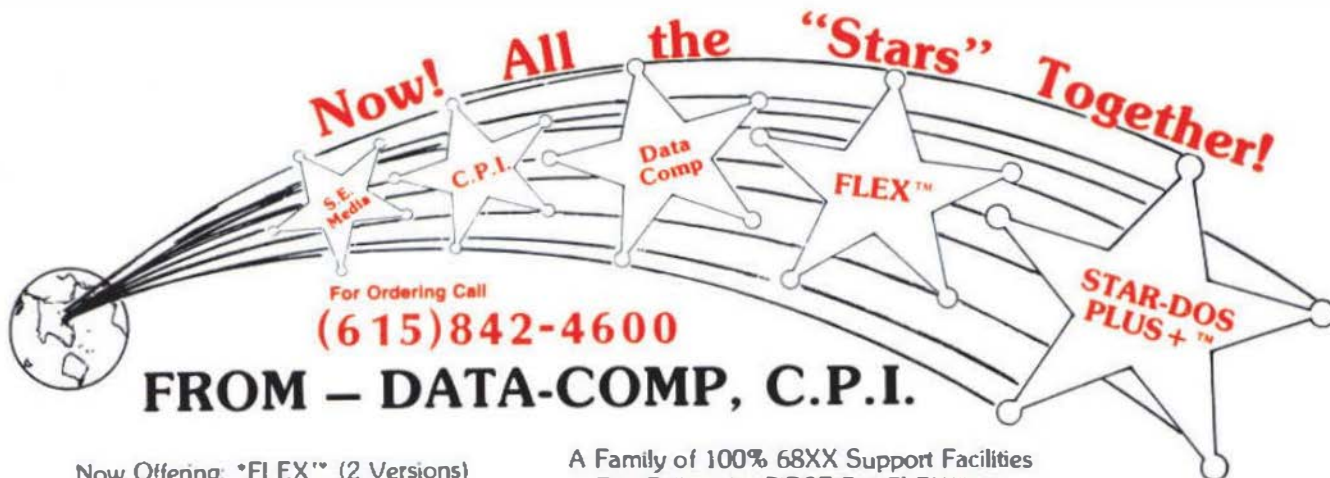
2 or more \$39.95 each

Add: \$7.50 S/H each

5900 Cassandra Smith Rd., Houston, Tx. 37343

Telephone 615 842-4600

Telex 510 600-6630



Now Offering: *FLEX* (2 Versions)
AND *STAR-DOS PLUS+™

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler

Complete with Manuals
Reg. \$250.⁰⁰ **Only \$79.⁰⁰**

STAR-DOS PLUS+

- Functions Same as FLEX
- Reads - writes FLEX Disks **\$34.⁰⁰**
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

FLEX-CoCo Jr.
without TSC
Editor & Assembler
\$49.⁰⁰

PLUS

ALL VERSIONS OF FLEX & STAR-DOS INCLUDE

TSC Editor

Reg \$50.00

NOW \$35.00

- + Read-Write-Dir RS Disk
- + Run RS Basic from Both
- + More Free Utilities

- + External Terminal Program
- + Test Disk Program
- + Disk Examine & Repair Program
- + Memory Examine Program
- + Many Many More!!!

TSC Assembler

Reg \$50.00

NOW \$35.00

CoCo Disk Drive Systems

2 THINLINE DOUBLE SIDED DOUBLE DENSITY DISK DRIVES
SYSTEM WITH POWER SUPPLY, CABINET, DISK DRIVE CABLE, J&M
NEW DISK CONTROLLER JPD-CP WITH J-DOS, RS-DOS OPERATING
SYSTEMS. **\$469.95**

* Specify What CONTROLLER You Want J&M, or RADIO SHACK

THINLINE DOUBLE SIDED
DOUBLE DENSITY 40 TRACKS **\$129.95**

Verbatim Diskettes

Single Sided Double Density **\$ 24.00**
Double Sided Double Density **\$ 24.00**

Controllers

J&M JPD-CP WITH J-DOS **\$139.95**
WITH J-DOS, RS-DOS **\$159.95**
RADIO SHACK 1.1 **\$134.95**

RADIO SHACK Disk CONTROLLER 1.1 **\$134.95**

Disk Drive Cables

Cable for One Drive **\$ 19.95**
Cable for Two Drives **\$ 24.95**

MISC

64K UPGRADE **\$ 29.95**
FOR C,D,E,F, AND COCO 11
RADIO SHACK BASIC 1.2 **\$ 24.95**
RADIO SHACK DISK BASIC 1.1 **\$ 24.95**

DISK DRIVE CABINET FOR A
SINGLE DRIVE **\$ 49.95**
DISK DRIVE CABINET FOR TWO
THINLINE DRIVES **\$ 69.95**

PRINTERS

EPSON LX-80 **\$289.95**
EPSON MX-70 **\$125.95**
EPSON MX-100 **\$495.95**

ACCESSORIES FOR EPSON

8148 2K SERIAL BOARD **\$ 89.95**
8149 32K EXPAND TO 128K **\$169.95**
EPSON MX-RX-80 RIBBONS **\$ 7.95**
EPSON LX-80 RIBBONS **\$ 5.95**
TRACTOR UNITS FOR LX-80 **\$ 39.95**
CABLES & OTHER INTERFACES
CALL FOR PRICING

DATA-COMP

5900 Cassandra Smith Rd.
Hixson, TN 37343



SHIPPING
USA ADD 2%
FOREIGN ADD 5%
MIN. \$2.50

(615)842-4600

For Ordering
Telex 5108008630

An Ace of a System in Spades! The New

MUSTANG-08/A™

Now with 4 serial ports standard & speed increase to 12 Mhz CPU + on board battery backup and includes the PROFESSIONAL OS-9 package - including the \$500.00 OS-9 C compiler! This offer won't last forever!

NOT 128K, NOT 512K FULL 768K No Wait RAM

The MUSTANG-08™ system took every hand from all other 68008 systems we tested, running OS-9 68K!

The MUSTANG-08 includes OS9-88K™ and/or Peter Stark's SKDOS™. SKDOS is a single user, single tasking system that takes up where "FLEX" left off. SKDOS is actually a 68XXX FLEX type system (Not a TSC product.)

The OS-9 68K system is a full blown multi-user, multi-tasking 68XXX system. All the popular 68000 OS-9 software runs. It's a speed whiz on disk I/O. Fact is, the MUSTANG-08 is faster on disk access than some other 68XXX systems are on memory cache access. Now, that is fast! And that's just a small part of the story! See benchmarks.

System includes OS-9 68K or SKDOS - Your Choice Specifications:

CPU	MC68008	12 Mhz
RAM	768K	256K Chips
	No Wait States	
PORTS	4 - RS232	MC88B1 QUART
	2 - 8 bit Parallel	MC8821 PIA
CLOCK	MK48T02	Real Time Clock Bat. B/U
EPROM	16K, 32K or 64K	Selectable
FLOPPY	WD1772	5 1/4 Drives
HARD DISK	Interface Port	WD1002 Board

Now more serial ports - faster CPU
Battery B/U - and \$850.00 OS-9 Profes-
sional with C compiler included!

*\$400.00

See Mustang-02 Ad - page 5
for trade-in details



MUSTANG-08

LOOK

Seconds 32 bit Register
Integer Long

Other 68008 8 Mhz OS-9 68K...18.0...9.0

MUSTANG-08 10 Mhz OS-9 68K...9.8...6.3

Main()

{

C Benchmark Loop

```
/* Init I; */
register long I;
for (I=0; I < 999999; ++I);
```

}

Now even faster!
with 12 Mhz CPU

C Compile times: OS-9 68K Hard Disk	
MUSTANG-08 8 Mhz CPU	0 min - 32 sec
Other popular 68008 system	1 min - 05 sec
MUSTANG-020	0 min - 21 sec



25 Megabyte
Hard Disk System

\$2,398.90

Complete with PROFESSIONAL OS-9
includes the \$500.00 C compiler, PC
style cabinet, heavy duty power supply,
5' DDDS 80 track floppy, 25 MegByte
Hard Disk - Ready to Run

Unlike other 68008 systems there are several significant differences. The MUSTANG-08 is a full 12 Megahertz system. The RAM uses NO wait states, this means full bore MUSTANG type performance.

Also, allowing for addressable ROM/PROM the RAM is the maximum allowed for a 68008. The 68008 can only address a total of 1 Megabytes of RAM. The design allows all the RAM space (for all practical purposes) to be utilized. What is not available to the user is required and reserved for the system.

A RAM disk of 480K can be easily configured, leaving 288K free for program/system RAM space. The RAM DISK can be configured to any size your application requires (system must have 128K in addition to its other requirements). Leaving the remainder of the original 768K for program use. Sufficient source included (drivers, etc.)

FLEX is a trademark of TSC

MUSTANG-08 is a trademark of CPI

Data-Comp Division



A Decade of Quality Service™

Systems World-Wide

Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, TN 37343

* Those with SW/PC hi-density FLEX 5' - Call for special info.